(19) Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 102 97 596 T5** 2004.12.02

(12) # Veröffentlichung

der internationalen Anmeldung mit der
(87) Veröffentlichungs-Nr.: **WO 03/058447**
in deutscher Übersetzung (Art. III § 8 Abs. 2 IntPatÜG)
(21) Deutsches Aktenzeichen: **102 97 596.5**
(86) PCT-Aktenzeichen: **PCT/US02/39786**
(86) PCT-Anmeldetag: **11.12.2002**
(87) PCT-Veröffentlichungstag: **17.07.2003**
(43) Veröffentlichungstag der PCT Anmeldung
in deutscher Übersetzung: **02.12.2004**

(51) Int Cl.[7]: **G06F 9/46**

(30) Unionspriorität:
10/039,579 **31.12.2001** **US**

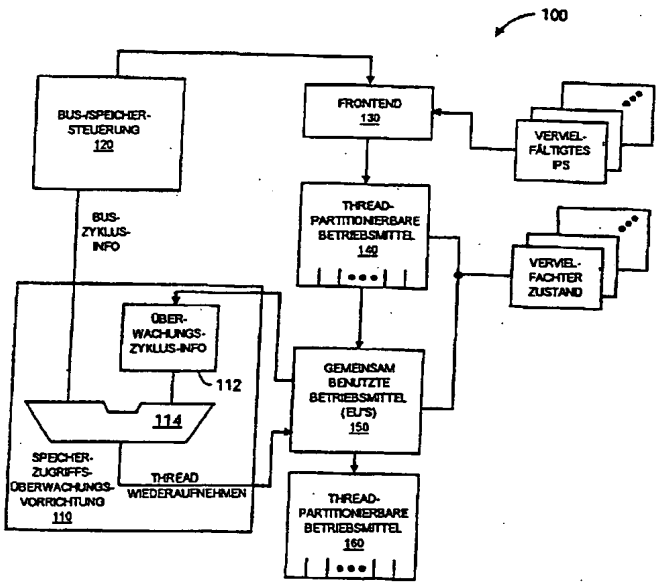
(71) Anmelder:
Intel Corporation, Santa Clara, Calif., US

(74) Vertreter:
BOEHMERT & BOEHMERT, 80336 München

(72) Erfinder:
- **Marr, Deborah, Portland, Oreg., US; Rodgers, Scott, Hillsboro, Oreg., US; Hill, David, Cornelius, Oreg., US; Kaushik, Shivandan, Portland, Oreg., US; Crossland, James, Banks, Oreg., US; Koufaty, David, Portland, Oreg., US**

(54) Bezeichnung: **Verfahren und Vorrichtung zum Suspendieren der Ausführung eines Threads, bis ein spezifizierter Speicherzugriff auftritt**

(57) Hauptanspruch: Prozessor, umfassend:
mehrere Ausführungseinheiten, um die Ausführung mehrerer Threads zu ermöglichen, einschließlich eines ersten Threads, wobei der erste Thread eine erste Anweisung mit einem zugeordneten Adressenoperanden, der eine Überwachungsadresse angibt, aufweist;
Suspendierungslogik zum Suspendieren der Ausführung des ersten Threads;
eine Überwachungsvorrichtung zur Bewirkung der Wiederaufnahme des ersten Threads als Reaktion auf einen Speicherzugriff auf die Überwachungsadresse.



Beschreibung

VERWANDTE ANMELDUNGEN

[0001] Die vorliegende Anmeldung ist mit der Anmeldung Nr. __/__,__, mit dem Titel "Suspending Execution of a Thread in a Multi-threaded Processor"; der Anmeldung Nr. __/__,__, mit dem Titel "Coherency Techniques for Suspending Execution of a Thread Until a Specified Memory Access Occurs"; Anmeldung Nr. __/__,__, mit dem Titel "Instruction Sequences for Suspending Execution of a Thread Until a Specified Memory Access Occurs", verwandt, die alle am selben Datum wie die vorliegende Anmeldung registriert wurden.

HINTERGRUND 1. Technisches Gebiet

[0002] Die vorliegende Offenlegung betrifft das Gebiet der Prozessoren und insbesondere Mehrfach-Thread-Prozessoren und Techniken zum Vorübergehenden Suspendieren der Verarbeitung eines Threads in einem Mehrfach-Thread-Prozessor.

2. Allgemeiner Stand der Technik

[0003] Ein Mehrfach-Thread-Prozessor kann gleichzeitig mehrere verschiedene Anweisungssequenzen verarbeiten. Ein Hauptmotivierungsfaktor für die Ausführung mehrerer Anweisungsströme in einem einzigen Prozessor ist die resultierende Verbesserung der Prozessorausnutzung. Über die Jahre haben sich parallele Architekturen entwickelt, es ist aber häufig schwierig, genug Parallelität aus einem einzigen Anweisungsstrom zu extrahieren, um die mehrfachen Ausführungseinheiten auszunutzen. Durch Prozessoren mit gleichzeitigen Mehrfach-Threads können mehrere Anweisungsströme gleichzeitig in den verschiedenen Ausführungsbetriebsmitteln ausgeführt werden, um zu versuchen, diese Betriebsmittel besser auszunützen. Mehrfach-Threads können besonders für solche Programme vorteilhaft sein, bei denen es zu Verzögerungen hoher Latenz kommt oder die häufig auf das Eintreten von Ereignissen warten. Wenn ein Thread darauf wartet, daß eine Task mit hoher Latenz fertig wird, oder auf ein bestimmtes Ereignis, kann ein anderer Thread verarbeitet werden.

[0004] Es wurden viele verschiedene Techniken vorgeschlagen, um zu steuern, wann ein Prozessor zwischen Threads wechselt. Zum Beispiel erkennen bestimmte Prozessoren bestimmte Ereignisse mit langer Latenz, wie zum Beispiel L2-Cache-Fehlspeicherungen und wechseln als Reaktion auf diese erkannten Ereignisse mit langer Latenz Threads. Obwohl die Erkennung solcher Ereignisse mit langer Latenz in bestimmten Umständen effektiv sein kann, erkennt eine solche Ereigniserkennung nur unwahrscheinlich alle Punkte, an denen es effizient sein kann, Threads zu wechseln. Insbesondere kann es sein, daß das Thread-Wechseln auf Ereignisbasis Punkte in einem Programm, an denen Verzögerungen vom Programmierer beabsichtigt sind, nicht erkennt.

[0005] Tatsächlich ist häufig der Programmierer am besten in der Lage, zu bestimmen, wann es effizient wäre, Threads zu wechseln, um verschwenderische Spin-Wait-Schleifen oder andere Betriebsmittel verbrauchende Verzögerungstechniken zu vermeiden. Indem Programmen erlaubt wird, den Thread-Wechsel zu steuern, können Programme also effizienter operieren. Zu diesem Zweck können explizite Programmanweisungen vorteilhaft sein, die sich auf die Thread-Auswahl auswirken. Zum Beispiel wird in der US-Patentanmeldung Nr. 09/489,130, registriert am 21.1.2000, eine „Pause"-Anweisung beschrieben. Durch die Pause-Anweisung kann ein Ausführungs-Thread vorübergehend suspendiert werden, bis entweder ein Zählwert erreicht wird oder bis eine Anweisung die Prozessor-Pipeline durchlaufen hat. Verschiedene Techniken können nützlich sein, indem Programmierern erlaubt wird, die Betriebsmittel eines Mehrfach-Thread-Prozessors effizienter einzuspannen.

Kurze Beschreibung der Zeichnungen

[0006] Die vorliegende Erfindung wird in den Figuren der beigefügten Zeichnungen als Beispiel und nicht als Beschränkung veranschaulicht.

[0007] **Fig.** 1 zeigt eine Ausführungsform eines Mehrfach-Thread-Prozessors mit einer Überwachungsvorrichtung zum Überwachen von Speicherzugriffen.

[0008] **Fig.** 2 ist ein Flußdiagramm der Funktionsweise des Mehrfach-Thread-Prozessors von **Fig.** 1 gemäß einer Ausführungsform.

[0009] **Fig.** 3 zeigt weitere Einzelheiten einer Ausführungsform eines Mehrfach-Thread-Prozessors.

[0010] **Fig.** 4 zeigt Betriebsmittel-Partitionierung-Sharing und -Duplikation gemäß einer Ausführungsform.

[0011] **Fig.** 5 ist ein Flußdiagramm des Suspendierens und Wiederaufnehmens der Ausführung eines Threads gemäß einer Ausführungsform.

[0012] **Fig.** 6a ist ein Flußdiagramm der Aktivierung und Funktionsweise von Überwachungslogik gemäß einer Ausführungsform.

[0013] **Fig.** 6b ist ein Flußdiagramm der Erweiterung der Beobachtbarkeit von Schreiboperationen gemäß einer Ausführungsform.

[0014] **Fig.** 7 ist ein Flußdiagramm von Überwachungsoperationen gemäß einer Ausführungsform.

[0015] **Fig.** 8 zeigt ein System gemäß einer Ausführungsform.

[0016] **Fig.** 9a–9c zeigen verschiedene Ausführungsformen von Softwaresequenzen, die offengelegte Prozessoranweisungen und -techniken verwenden.

[0017] **Fig.** 10 zeigt eine alternative Ausführungsform, die es ermöglicht, daß eine überwachte Adresse Cache-gespeichert bleibt.

[0018] **Fig.** 11 zeigt die verschiedenen Entwurfsrepräsentationen oder -formate zur Simulation, Emulation und Herstellung eines Entwurfs unter Verwendung der offengelegten Techniken.

Ausführliche Beschreibung

[0019] Die folgende Beschreibung beschreibt Techniken zum Suspendieren der Ausführung eines Threads bis ein spezifizierter Speicherzugriff auftritt. In der folgenden Beschreibung werden zahlreiche spezifische Einzelheiten, wie zum Beispiel logische Implementierungen, Opcodes, Mittel zum Spezifizieren von Operanden, Implementierungen für Betriebsmittel-Partitionierung/-Sharing/-Duplikation, Typen und Beziehungen von Systemkomponenten und Wahlmöglichkeiten für logische Partitionierung/Integration dargelegt, um ein besseres Verständnis der vorliegenden Erfindung zu ermöglichen. Für Fachleute ist jedoch erkennbar, daß die Erfindung ohne solche spezifischen Einzelheiten ausgeübt werden kann. In anderen Fällen wurden Steuerstrukturen, Schaltung auf Gatterebene und volle Softwareanweisungssequenzen nicht im Einzelnen gezeigt, um die Erfindung nicht zu verdecken. Anhand der angegebenen Beschreibungen werden Durchschnittsfachleute in der Lage sein, ohne übermäßiges Experimentieren geeignete Funktionalität zu implementieren.

[0020] Durch die offengelegten Techniken kann ein Programmierer einen Wartemechanismus in einem Thread implementieren, während andere Threads Verarbeitungsbetriebsmittel einspannen können. Es kann eine Überwachungsvorrichtung eingerichtet werden, so daß ein Thread suspendiert werden kann, bis ein bestimmter Speicherzugriff, wie zum Beispiel eine Schreiboperation in eine spezifizierte Speicherstelle auftritt. Somit kann ein Thread bei einem spezifizierten Ereignis wiederaufgenommen werden, ohne eine prozessorbetriebsmittelverschwendende Routine wie etwa eine Spin-Wait-Schleife auszuführen. Bei bestimmten Ausführungsformen können zuvor dem suspendierten Thread zugeordnete Partitionen freigegeben werden, während der Thread suspendiert ist. Diese und/oder andere offengelegte Techniken können vorteilhafterweise den Gesamtprozessordurchsatz verbessern.

[0021] **Fig.** 1 zeigt eine Ausführungsform eines Mehrfach-Thread-Prozessors **100** mit einer Speicherzugriffüberwachungsvorrichtung **110** zur Überwachung von Speicherzugriffen. Bei bestimmten Ausführungsformen kann ein „Prozessor" als eine einzige integrierte Schaltung gebildet werden. Bei anderen Ausführungsformen können mehrere integrierte Schaltungen zusammen einen Prozessor bilden, und bei noch anderen Ausführungsformen können Hardware- und Softwareroutinen (z.B. binäre Übersetzungsroutinen) zusammen den Prozessor bilden. Bei der Ausführungsform von **Fig.** 1 führt eine Bus-/Speichersteuerung **120** einem Frontend **130** auszuführende Anweisungen zu. Das Frontend **130** lenkt das Abrufen von Anweisungen von verschiedenen Threads gemäß Anweisungszeigern **170**. Anweisungszeigerlogik ist vervielfältigt, um mehrere Threads zu unterstützen.

[0022] Das Frontend **130** leitet Anweisungen in Thread-partitionierbare Betriebsmittel **140** zur weiteren Verarbeitung. Die Thread-partitionierbaren Betriebsmittel **140** enthalten logisch getrennte Partitionen, die fest bestimmten Threads zugeordnet sind, wenn mehrere Threads in dem Prozessor **100** aktiv sind. Bei einer Ausführungsform enthält jede getrennte Partition nur Anweisungen aus dem Thread, dem dieser Teil fest zugeordnet ist. Die Thread-partitionierbaren Betriebsmittel **140** können zum Beispiel Anweisungswarteschlangen enthalten. In einem Einzel-Thread-Modus können die Partitionen der Thread-partitionierbaren Betriebsmittel **140** kombiniert werden, um eine einzige große Partition zu bilden, die dem einen Thread fest zugeordnet ist.

[0023] Außerdem enthält der Prozessor **100** den vervielfältigten Zustand **180**. Der vervielfältigte Zustand **180** enthält Zustandsvariablen, die ausreichen, um den Kontext für einen logischen Prozessor zu halten. Mit dem vervielfältigten Zustand **180** können mehrere Threads ausgeführt werden, ohne um Zustandsvariablenspeicherung zu konkurrieren. Zusätzlich kann für jeden Thread Registerzuteilungslogik vervielfältigt sein. Die vervielfältigte zustandsbezogene Logik operiert mit den entsprechenden Betriebsmittelpartitionen, um ankommende Anweisungen für die Ausführung vorzubereiten.

[0024] Die Thread-partitionierbaren Betriebsmittel **140** leiten Anweisungen zu gemeinsam benutzten Betriebsmitteln **150** weiter. Die gemeinsam benutzten Betriebsmittel **150** operieren an Anweisungen ungeachtet ihres Ursprungs. Zum Beispiel können Scheduler- und Ausführungseinheiten Thread-unbewußte gemeinsam benutzte Betriebsmittel sein. Die partitionierbaren Betriebsmittel **140** können den ge-

meinsam benutzten Betriebsmitteln **150** Anweisungen aus mehreren Threads zuführen, indem zwischen den Threads auf eine faire Weise alterniert wird, die einen fortgesetzten Fortschritt an jedem aktiven Thread bereitstellt. Somit können die gemeinsam benutzten Betriebsmittel die bereitgestellten Anweisungen an dem entsprechenden Zustand ohne Sorge um die Thread-Mischung ausführen.

[0025] Den gemeinsam benutzten Betriebsmitteln **150** kann eine weitere Menge von Thread-partitionierbaren Betriebsmitteln **160** folgen. Die Thread-partitionierbaren Betriebsmittel **160** können Ausscheidungsbetriebsmittel wie zum Beispiel ein Umordnungspuffer und dergleichen enthalten. Folglich können die Thread-partitionierbaren Betriebsmittel **160** sicherstellen, daß die Ausführung von Anweisungen aus jedem Thread ordnungsgemäß endet, und daß der entsprechende Zustand für diesen Thread entsprechend aktualisiert wird.

[0026] Wie bereits erwähnt, kann es wünschenswert sein, Programmierern eine Technik zur Implementierung der Funktionalität einer Spin-Wait-Schleife bereitzustellen, ohne ein konstantes Abfragen einer Speicherstelle oder sogar Ausführen von Anweisungen zu erfordern. Somit enthält der Prozessor **100** von **Fig.** 1 die Speicherzugriffsüberwachungsvorrichtung **110**. Die Speicherzugriffsüberwachungsvorrichtung **110** kann mit Informationen über einen Speicherzugriffszyklus programmiert werden, für dessen Beobachtung die Überwachungsvorrichtung **110** freigegeben werden kann. Folglich enthält die Überwachungsvorrichtung **110** ein Überwachungsvorrichtungszyklusinformationsregister **112**, das durch Vergleichslogik **114** mit aus der Bus-/Speichersteuerung **120** empfangenen Buszyklusinformationen verglichen wird. Wenn es zu einer Übereinstimmung kommt, wird ein Thread-Wiederaufnehmen-Signal erzeugt, um einen suspendierten Thread neu zu starten. Aus internen und/oder externen Bussen des Prozessors können Speicherzugriffsinformationen erhalten werden.

[0027] Das Überwachungsvorrichtungszyklusinformationsregister **112** kann Einzelheiten enthalten, die die Art des Zyklus und/oder die Adresse, die die Wiederaufnahme eines Threads auslösen sollte, spezifizieren. Bei einer Ausführungsform speichert das Überwachungsvorrichtungszyklusinformationsregister **112** eine physikalische Adresse und die Überwachungsvorrichtung wartet auf jegliche Buszyklen, die ein tatsächliches oder potentielles Schreiben in diese physikalische Adresse anzeigen. Ein solcher Zyklus kann in Form eines expliziten Schreibzyklus vorliegen und/oder kann eine Leseoperation für Eigentümerschaft oder ein invalidierender Zyklus durch einen anderen Agenten sein, der versucht, die exklusive Eigentümerschaft einer Cache-speicherbaren Leitung zu übernehmen, so daß er ohne eine externe Bustransaktion auf diese Leitung schreiben kann. In jedem Fall kann die Überwachungsvorrichtung so programmiert werden, daß sie bei verschiedenen Ausführungsformen an verschiedenen Transaktionen ausgelöst wird.

[0028] Die Operationen der Ausführungsform von **Fig.** 1 können mit Bezug auf das Flußdiagramm von **Fig.** 2 weiter erläutert werden. Bei einer Ausführungsform enthält der Anweisungssatz des Prozessors **100** einen MONITOR-Opcode (Anweisung), der die Überwachungstransaktionsinformationen einrichtet. Im Block **200** wird der MONITOR-Opcode als Teil der Anweisungssequenz eines ersten Threads (T1) empfangen. Wie im Block **210** angegeben, ermöglicht es der Prozessor **100** als Reaktion auf den MONITOR-Opcode der Überwachungsvorrichtung **110**, Speicherzugriffe für den spezifizierten Speicherzugriff zu überwachen. Der auslösende Speicherzugriff kann durch einen impliziten oder einen expliziten Operanden spezifiziert werden. Deshalb kann das Ausführen des MONITOR-Opcodes die Überwachungsadresse spezifizieren, da die Überwachungsadresse im Voraus in einem Register oder in einer anderen Speicherstelle als ein impliziter Operand gespeichert werden kann. Wie im Block **215** angegeben, prüft die Überwachungsvorrichtung, ob der spezifizierte Zyklus erkannt wurde. Wenn nicht, überwacht die Überwachungsvorrichtung weiter Speicherzugriffe. Wenn der auslösende Zyklus erkannt wird, wird ein Anstehendes-Überwachungsereignis-Anzeiger gesetzt, wie im Block **220** angegeben.

[0029] Die Ausführung des MONITOR-Opcodes löst die Aktivierung der Überwachungsvorrichtung **110** aus. Die Überwachungsvorrichtung **110** kann beginnen, parallel mit anderen Operationen in dem Prozessor zu operieren. Bei einer Ausführungsform richtet die MONITOR-Anweisung selbst nur die Überwachungsvorrichtung **110** mit den ordnungsgemäßen Speicherzyklusinformationen ein und aktiviert die Überwachungsvorrichtung **110**, ohne Überwachungsereignisse zu entmaskieren. Anders ausgedrückt, können sich bei dieser Ausführungsform nach der Ausführung des MONITOR-Opcodes Überwachungsereignisse ansammeln, was jedoch unerkannt bleiben kann, wenn sie nicht explizit entmaskiert werden.

[0030] Im Block **225** wird also das Auslösen eines Speicherwartens als ein separates Ereignis angegeben. Bei bestimmten Ausführungsformen kann ein Opcode für Speicherwarten (MWAIT) verwendet werden, um die Erkennung von Überwachungsereignissen und die Suspendierung von T1 auszulösen. Durch die Verwendung zweier getrennter Anweisungen zum Einrichten und zum Auslösen der Thread-Suspendierung kann ein Programmierer zusätzliche Flexibilität erhalten und es kann ein effizienteres Programmieren möglich werden. Eine alternati-

ve Ausführungsform löst das Speicherwarten jedoch von dem ersten Opcode aus, der auch die Überwachungsvorrichtung **110** einrichtet. In jedem Fall aktivieren eine oder mehrere Anweisungen die Überwachungsvorrichtung und ermöglichen das Erkennen von Überwachungsereignissen.

[0031] Bei Ausführungsformen, bei denen getrennte Opcodes zum Aktivieren der Überwachungsvorrichtung **110** und zum Auslösen des Erkennens von Überwachungsereignissen verwendet werden, kann es vorteilhaft sein, eine Prüfung durchzuführen, um sicherzustellen, daß die Überwachungsvorrichtung aktiviert worden ist, bevor der Thread suspendiert wird, wie im Block **230** gezeigt. Durch Prüfen, ob ein Überwachungsereignis bereits ansteht (nicht gezeigt) kann zusätzlich die Suspendierung von T1 vermieden werden und die Operationen können im Block **250** fortgesetzt werden. Unter der Annahme, daß die Überwachungsvorrichtung **110** freigegeben wurde und keine Überwachungsereignisse bereits anstehen, kann T1 suspendiert werden, wie im Block **235** gezeigt.

[0032] Mit suspendiertem T1 tritt der Prozessor in einen implementierungsabhängigen Zustand ein, der es anderen Threads ermöglicht, die Prozessorbetriebsmittel besser auszunutzen. Bei bestimmten Ausführungsformen kann der Prozessor einen Teil der Partitionen der partitionierbaren Betriebsmittel **140** und **160**, die fest T1 zugeordnet waren, oder alle diese freigeben. Bei anderen Ausführungsformen können verschiedene Permutationen des MONITOR-Opcodes oder von diesem zugeordneten Einstellungen angeben, welche etwaigen Betriebsmittel freigegeben werden. Wenn zum Beispiel ein Programmierer ein kürzeres Warten antizipiert, kann der Thread suspendiert werden, aber seine Betriebsmittelpartitionen aufrechterhalten. Der Durchsatz ist immer noch verbessert, da die gemeinsam benutzten Betriebsmittel während des Thread-Suspendierungszeitraums ausschließlich von anderen Threads verwendet werden können. Wenn ein längeres Warten antizipiert wird, können durch Freigabe aller dem suspendierten Thread zugeordneten Partitionen andere Threads zusätzliche Betriebsmittel erhalten, wodurch der Durchsatz der anderen Threads potentiell erhöht wird. Der zusätzliche Durchsatz wird jedoch zu Lasten des Overheads erhalten, das dem Entfernen und Hinzufügen von Partitionen, wenn Threads suspendiert bzw. wiederaufgenommen werden, zugeordnet ist.

[0033] T1 bleibt in einem suspendierten Zustand, bis ein Überwachungsereignis ansteht. Wie bereits besprochen, operiert die Überwachungsvorrichtung **110** unabhängig, um Überwachungsereignisse zu erkennen und zu signalisieren (Blöcke **215–220**). Wenn der Prozessor im Block **240** erkennt, daß ein Überwachungsereignis ansteht, wird T1 wiederaufgenommen, wie im Block **250** angegeben. Es muß keine aktive Verarbeitung von Anweisungen in T1 auftreten, damit das Überwachungsereignis T1 aufweckt. Statt dessen bleibt T1 suspendiert und die freigegebene Überwachungsvorrichtung **110** signalisiert dem Prozessor ein Ereignis. Der Prozessor wickelt das Ereignis ab, erkennt, daß das Ereignis anzeigt, daß T1 wiederaufgenommen werden soll, und führt die entsprechenden Aktionen zur Wiederaufnahme von T1 durch.

[0034] Die Ausführungsformen von **Fig.** 1 und 2 liefern also Techniken, durch die ein durch ein Programm suspendierter Thread nach dem Auftreten eines spezifizierten Speicherzugriffs wiederaufgenommen werden kann. Bei einer Ausführungsform bewirken auch andere Ereignisse, daß **71** wiederaufgenommen wird. Zum Beispiel kann ein Interrupt bewirken, daß T1 wiederaufgenommen wird. Durch eine solche Implementierung ist es vorteilhafterweise möglich, daß die Überwachungsvorrichtung insofern weniger als perfekt ist, als sie bestimmte Speicherzugriffe oder andere Bedingungen, die eine Wiederaufnahme des Threads verursachen sollten, verfehlt (nicht erkennt). Folglich kann T1 manchmal unnötigerweise aufgeweckt werden. Eine solche Implementierung verringert jedoch die Wahrscheinlichkeit, daß T1 aufgrund eines verfehlten Ereignisses permanent eingefroren wird, wodurch Hardwareentwurf und -validierung vereinfacht werden. Die unnötigen Erweckungen von T1 können nur eine geringfügige Unannehmlichkeit sein, da eine Schleife konstruiert werden kann, so daß T1 nachprüft, ob die Bedingung, die er erwartet hat, wirklich aufgetreten ist, und er sich nochmals selbst suspendieren kann, wenn dies nicht der Fall ist.

[0035] Bei bestimmten Ausführungsformen können die Thread-partitionierbaren Betriebsmittel, die vervielfältigten Betriebsmittel und die gemeinsam benutzten Betriebsmittel verschieden angeordnet sein. Bei bestimmten Ausführungsformen liegen möglicherweise nicht an beiden Enden der gemeinsam benutzten Betriebsmittel partitionierbare Betriebsmittel vor. Bei bestimmten Ausführungsformen können die partitionierbaren Betriebsmittel möglicherweise nicht strikt partitioniert sein, sondern statt dessen bestimmten Anweisungen ermöglichen, Partitionen zu überschreiten, oder Partitionen ermöglichen, eine verschiedene Größe aufzuweisen, abhängig von dem in dieser Partition ausgeführten Thread oder von der Gesamtzahl von ausgeführten Threads. Außerdem können verschiedene Mischungen von Betriebsmitteln als gemeinsam benutzte, duplizierte und partitionierte Betriebsmittel ausgewiesen werden.

[0036] **Fig.** 3 zeigt weitere Einzelheiten einer Ausführungsform eines Mehrfach-Thread-Prozessors. Die Ausführungsform von **Fig.** 3 enthält unter anderem mit Kohärenz zusammenhängende Logik **350**,

eine Implementierung einer Überwachungsvorrichtung **310** und eine spezifische Implementierung von Thread-Suspendierungs- und -wiederaufnahmelogik **377**. Bei der Ausführungsform von **Fig.** 3 enthält eine Busschnittstelle **300** eine Bussteuerung **340**, Ereigniserkennungslogik **345**, eine Überwachungsvorrichtung **310** und die mit Kohärenz zusammenhängende Logik **350**.

[0037] Die Busschnittstelle **300** führt einem Frontend **365** Anweisungen zu, das die Erzeugung von Mikrooperanden (uOP) durchführt und aus Makroanweisungen uOPs erzeugt. Die Ausführungsbetriebsmittel **370** empfangen uOPS von dem Frontend **365**, und Backend-Logik **380** zieht die verschiedenen uOPs zurück, nachdem sie ausgeführt wurden. Bei einer Ausführungsform unterstützen das Frontend, das Backend und Ausführungsbetriebsmittel eine Ausführung außerhalb der Reihenfolge.

[0038] Mit Bezug auf **Fig.** 5–9 werden verschiedene Einzelheiten von Operationen weiter besprochen. Kurz gefaßt kann jedoch ein MONITOR-Opcode durch die Busschnittstelle **300** in den Prozessor eintreten und durch das Frontend **365** für die Ausführung vorbereitet werden. Bei einer Ausführungsform wird zur Ausführung durch die Ausführungsbetriebsmittel **370** ein spezieller MONITOR-uOP erzeugt. Der MONITOR-uOP kann von den Ausführungseinheiten ähnlich wie eine Speicheroperation behandelt werden, wobei die Überwachungsadresse durch Adressenübersetzungslogik **375** in eine physikalische Adresse übersetzt wird, die der Überwachungsvorrichtung **310** zugeführt wird. Die Überwachungsvorrichtung **310** kommuniziert mit Thread-Suspendierungs- und -wiederaufnahmelogik **377**, um die Wiederaufnahme von Threads zu bewirken. Die Thread-Suspendierungs- und -wiederaufnahmelogik kann Partition und Verschmelzung von Betriebsmitteln durchführen, wenn sich die Anzahl aktiver Threads ändert.

[0039] Zum Beispiel zeigt **Fig.** 4 das Partitionieren, Duplizieren und gemeinsame Benutzen von Betriebsmitteln gemäß einer Ausführungsform. Partitionierte Betriebsmittel können gemäß dem Auf und Ab aktiver Threads in der Maschine partitioniert und verschmolzen (zur Wiederverwendung durch andere Threads wieder zusammengeschmolzen) werden. Bei der Ausführungsform von **Fig.** 4 umfassen duplizierte Betriebsmittel Anweisungszeigerlogik in dem Anweisungsabrufteil der Pipeline, Registerumbenennungslogik in dem Umbenennungsteil der Pipeline, (nicht gezeigte, aber in verschiedenen Stufen in der Pipeline erwähnte) Zustandsvariablen und eine Interrupt-Steuerung (nicht gezeigt, im allgemeinen asynchron zur Pipeline). Gemeinsam benutzte Betriebsmittel in der Ausführungsform von **Fig.** 4 umfassen Scheduler in der Schedule-Stufe der Pipeline, ein Registerpool in den Register-Lese- und -schreibteilen der Pipeline und Ausführungsbetriebsmittel im Ausführungsteil der Pipeline. Zusätzlich können ein Trace-Cache und ein L1-Daten-Cache gemeinsam benutzte Betriebsmittel sein, die gemäß Speicherzugriffen ungeachtet des Thread-Kontexts aufgefüllt werden. Bei anderen Ausführungsformen kann bei Cache-Speicherungsentscheidungen Thread-Kontext berücksichtigt werden. Partitionierte Betriebsmittel in der Ausführungsform von **Fig.** 4 umfassen zwei Warteschlangen in Warteschlangenstufen der Pipeline, einen Umordnungspuffer in einer Ausscheidungsstufe der Pipeline und einen Speicherpuffer. Thread-Auswahlmultiplexlogik alterniert zwischen den verschiedenen duplizierten und partitionierten Betriebsmitteln, um beiden Threads einen sinnvollen Zugriff zu gewähren.

[0040] Als Beispiel wird angenommen, daß bei der weiteren Beschreibung der Funktionsweise einer Ausführungsform des Prozessors von **Fig.** 3, das in **Fig.** 4 gezeigte Partitionieren, gemeinsame Benutzen und Duplizieren in Verbindung mit der Ausführungsform von **Fig.** 3 verwendet wird. Insbesondere werden nun weitere Einzelheiten der Funktionsweise der Ausführungsform von **Fig.** 3 mit Bezug auf das Flußdiagramm von **Fig.** 5 besprochen. Es wird angenommen, daß der Prozessor in einem Mehrfach-Thread-Modus arbeitet, wobei mindestens zwei Threads aktiv sind.

[0041] Im Block **500** empfängt das Frontend **365** einen MONITOR-Opcode während der Ausführung eines ersten Threads (T1). Das Frontend **365** erzeugt bei einer Ausführungsform einen speziellen Überwachungs-uOP. Der MONITOR-uOP wird zu den Ausführungsbetriebsmitteln **370** weitergeleitet. Der Überwachungs-uOP weist eine zugeordnete Adresse auf, die die zu überwachende Adresse angibt (die Überwachungsadresse). Die zugeordnete Adresse kann in Form eines expliziten Operanden oder eines impliziten Operanden vorliegen (d.h. die zugeordnete Adresse ist einem vorbestimmten Register oder einer anderen Speicherstelle zu entnehmen). Durch die zugeordnete Adresse wird insofern die Überwachungsadresse „angegeben", als sie genug Informationen zur Bestimmung der Überwachungsadresse (möglicherweise in Verbindung mit anderen Registern oder Informationen) übermittelt. Zum Beispiel kann die zugeordnete Adresse eine lineare Adresse sein, die eine entsprechende physikalische Adresse aufweist, die die fragliche Überwachungsadresse ist. Alternativ dazu könnte die Überwachungsadresse in einem virtuellen Adressenformat gegeben oder als eine relative Adresse angegeben oder auf andere bekannte oder zweckmäßige Adressenspezifizierungsweisen spezifiziert werden. Wenn virtuelle Adressenoperanden verwendet werden, kann es wünschenswert sein, wenn allgemeine Programmfehler als Break-Ereignisse erkannt werden.

[0042] Die Überwachungsadresse kann jede beliebige zweckmäßige Speichereinheit für die Überwachung angeben. Zum Beispiel kann die Überwachungsadresse bei einer Ausführungsform eine Cache-Leitung angeben. Bei alternativen Ausführungsformen kann die Überwachungsadresse jedoch einen Teil einer Cache-Leitung, einen Teil einer spezifischen gewählten Größe oder eine Einheit von Speicher mit verschiedenen Beziehungen zu den Cache-Leitungsgrößen verschiedener Prozessoren oder eine einzige Adresse angeben. Die Überwachungsadresse kann also eine Einheit angeben, die durch den Operanden spezifizierte Daten (und weitere Daten) enthält, oder kann spezifisch eine Adresse für eine gewünschte Dateneinheit angeben.

[0043] Bei der Ausführungsform von **Fig.** 3 wird die Überwachungsadresse der Adressenübersetzungslogik **375** zugeführt und zu der Überwachungsvorrichtung **310** weitergeleitet und dort in einem Überwachungsadressenregister **335** gespeichert. Als Reaktion auf den MONITOR-Opcode wird die Überwachungsvorrichtung **310** dann durch die Ausführungsbetriebsmittel **370** freigegeben und aktiviert, wie im Block **510** angegeben und in **Fig.** 6 weiter erläutert. Wie später mit Bezug auf **Fig.** 6 weiter besprochen werden wird, kann es vorteilhaft sein, etwaige Speicheroperationen, die nach dem MONITOR-Opcode auftreten, einzusperren, um sicherzustellen, daß Speicheroperationen verarbeitet und deshalb erkannt werden, bevor jegliche Thread-Suspendierung auftritt. Es müssen möglicherweise also bei dieser Ausführungsform als Ergebnis des Aktivierens der Überwachungsvorrichtung **310** bestimmte Operationen auftreten, bevor jegliche nachfolgende Anweisungen unternommen werden können. Der Block **510** ist jedoch als parallel zu dem Block **505** auftretend gezeigt, weil bei dieser Ausführungsform die Überwachungsvorrichtung **310** weiter parallel mit anderen Operationen operiert, bis ein Break-Ereignis auftritt, nachdem sie durch den MONITOR-Opcode aktiviert wurde.

[0044] Im Block **505** wird in Thread **1** ein Opcode für Speicherwarten (MWAIT) empfangen und zur Ausführung weitergeleitet. Die Ausführung des MWAIT-Opcodes entmaskiert Überwachungsereignisse in der Ausführungsform von **Fig.** 5. Als Reaktion auf den MWAIT-Opcode wird wie im Block **515** angegeben geprüft, ob ein Überwachungsereignis ansteht. Wenn kein Überwachungsereignis ansteht, wird im Block **520** eine Prüfung durchgeführt, um sicherzustellen, daß die Überwachungsvorrichtung aktiv ist. Wenn zum Beispiel ein MWAIT ausgeführt wird, ohne daß zuvor ein MONITOR ausgeführt wurde, wäre die Überwachungsvorrichtung **310** nicht aktiv. Wenn entweder die Überwachungsvorrichtung inaktiv ist oder ein Überwachungsereignis ansteht, wird die Ausführung von Thread **1** im Block **580** fortgesetzt.

[0045] Wenn die Überwachungsvorrichtung **310** aktiv ist und kein Überwachungsereignis ansteht, dann wird die Ausführung von Thread **1** suspendiert, wie im Block **525** angegeben. Die Thread-Suspendierungs-/-wiederaufnahmelogik **377** enthält Pipeline-Flush-Logik **382**, die die Prozessorpipeline entleert, um alle Anweisungen auszuräumen, wie im Block **530** angegeben.

[0046] Nachdem die Pipeline entleert wurde, bewirkt die Partitions-/Verschmelzungslogik **385**, daß jegliche exklusiv Thread **1** zugeordnete Betriebsmittel zur Verwendung durch andere Threads freigegeben werden, wie im Block **535** angegeben. Diese freigegebenen Betriebsmittel werden verschmolzen, um zur Benutzung durch die übrigen aktiven Threads eine Menge größerer Betriebsmittel zu bilden. Zum Beispiel werden mit Bezug auf das Zweithreadbeispiel von **Fig.** 4 alle mit Thread **1** zusammenhängenden Anweisungen aus beiden Warteschlangen entleert. Jedes Paar von Warteschlange wird dann kombiniert, um dem zweiten Thread eine größere Warteschlange bereitzustellen. Ähnlich werden dem zweiten Thread weitere Register aus dem Registerpool zur Verfügung gestellt, weitere Einträge aus dem Speicherpuffer werden für den zweiten Thread befreit und weitere Einträge in dem Umordnungspuffer werden dem zweiten Thread zur Verfügung gestellt. Im wesentlichen werden diese Strukturen wieder in einzelne festzugeordnete Strukturen der doppelten Größe verwandelt. Natürlich können sich verschiedene Proportionen aus Implementierungen ergeben, die verschieden viele Threads verwenden.

[0047] In den Blöcken **540**, **545** und **550** werden verschiedene Ereignisse geprüft, um zu bestimmen, ob Thread **1** wieder aufgenommen werden soll. Es ist zu beachten, daß diese Prüfungen nicht durch Anweisungen durchgeführt werden, die als Teil von Thread **1** ausgeführt werden. Statt dessen werden diese Operationen durch den Prozessor parallel mit seiner Verarbeitung anderer Threads durchgeführt. Wie ausführlicher mit Bezug auf **Fig.** 6 besprochen werden wird, prüft die Überwachungsvorrichtung selbst, ob ein Überwachungs-Schreibereignis aufgetreten ist, und zeigt dies durch Setzen eines Anstehendes-Ereignis-Anzeigers an. Der Anstehendes-Ereignis-Anzeiger wird über eine EVENT-Signal der Suspendierungs-/Wiederaufnahmelogik **377** zugeführt (z.B. Mikrocode). Mikrocode kann bei einer Ausführungsform das Überwachungsereignis an einer entsprechenden Anweisungsgrenze erkennen (Block **540**), da dieses Ereignis im Block **505** durch den MWAIT-Opcode entmaskiert wurde. Die Ereignisdetektionslogik **345** kann andere Ereignisse erkennen, wie zum Beispiel Interrupts, die als Break-Ereignisse ausgewiesen sind (Block **545**). Zusätzlich kann ein optionaler Timer verwendet werden, um periodisch aus dem Speicherwartezustand auszutreten, um sicherzustellen, daß der Prozessor nicht aufgrund ei-

ner bestimmten Ereignissequenz einfriert (Block **550**). Wenn keine dieser Ereignisse einen Austritt für den Speicherwartezustand signalisieren, bleibt Thread **1** suspendiert.

[0048] Wenn Thread **1** wieder aufgenommen wird, wird die Thread-/Suspendierungswiederaufnahmelogik **377** nach Erkennung des entsprechenden Ereignisses nochmals aktiviert. Wieder wird die Pipeline ausgeräumt, wie im Block **560** angegeben, um Anweisungen aus der Pipeline zu entleeren, so daß Betriebsmittel nochmals partitioniert werden können, um den bald aufzuweckenden Thread **1** zu berücksichtigen. Im Block **570** werden die entsprechenden Betriebsmittel neu partitioniert und Thread **1** wird im Block **580** wiederaufgenommen.

[0049] **Fig.** 6a zeigt weitere Einzelheiten der Aktivierung und Funktionsweise der Überwachungsvorrichtung **310**. Im Block **600** wird das Frontend-Abrufen für Thread **1** gestoppt, um zu verhindern, daß weitere Operationen von Thread **1** in die Maschine eintreten. Im Block **605** wird der zugeordnete Adressenoperand durch die Adressenübersetzungslogik **375** von einer linearen Adresse in eine physikalische Adresse umgesetzt. Im Block **610** wird die Beobachtbarkeit von Schreiboperationen an die überwachte Adresse erhöht. Im allgemeinen besteht das Ziel dieser Operation darin, Cache-Agenten dazu zu zwingen, Schreib-Operationen, die sich auf an der Überwachungsadresse gespeicherte Informationen auswirken würden, der Überwachungsvorrichtung **310** selbst sichtbar zu machen. Weitere Einzelheiten einer spezifischen Implementierung werden mit Bezug auf **Fig.** 6b besprochen. Im Block **615** wird die physikalische Adresse zur Überwachung gespeichert, obwohl zu beachten ist, daß diese Adresse früher oder später in dieser Sequenz gespeichert werden kann.

[0050] Wie im Block **620** angegeben, wird als nächstes die Überwachungsvorrichtung freigegeben. Die Überwachungsvorrichtung überwacht Buszyklen auf Schreiboperationen in die physikalische Adresse, die die in dem Überwachungsadressenregister **335** gespeicherte Überwachungsadresse ist. Weitere Einzelheiten der Überwachungsoperation werden nachfolgend mit Bezug auf **Fig.** 7 besprochen. Nachdem die Überwachungsvorrichtung freigegeben wurde, wird wie im Block **625** angegeben eine Speichereinsperroperation ausgeführt. Das Speichereinsperren hilft dabei, sicherzustellen, daß alle Speicheroperationen in der Maschine dann verarbeitet werden, wenn der MONITOR-Opcode mit der Ausführung fertig ist. Wenn alle Speicheroperationen aus der Zeit, bevor MONITOR aus der Maschine entleert wird, ist die Wahrscheinlichkeit, daß fälschlich in einen Speicherwartezustand eingetreten wird, reduziert. Die Speichereinsperroperation ist jedoch eine Vorsichtsmaßnahme und kann eine zeitaufwendige Operation sein.

[0051] Dieses Speichereinsperren ist optional, weil der MONITOR/MWAIT-Mechanismus dieser Ausführungsform als Mehrfachaustrittsmechanismus ausgelegt wurde. Anders ausgedrückt, können auch verschiedene Ereignisse, wie zum Beispiel bestimmte Interrupts, System- oder Onboard-Timer usw. einen Austritt aus dem Speicherwartezustand verursachen. Es ist also bei dieser Ausführungsform nicht garantiert, daß der einzige Grund für ein Aufwecken des Threads darin besteht, daß sich der überwachte Datenwert verändert hat. Folglich (siehe auch **Fig.** 9a–c) sollte bei dieser Implementierung Software nachprüfen, ob sich der in dem Speicher gespeicherte bestimmte Wert verändert hat. Bei einer Ausführungsform sind bestimmte Ereignisse, darunter das Setzen von INTR-, NMI- und SMI-Interrupts; Maschinenprüf-Interrupts und Fehler Break-Ereignisse und andere, wie zum Beispiel Powerdown-Ereignisse, sind es nicht. Bei einer Ausführungsform ist das Setzen des A20M-Anschlusses auch ein Break-Ereignis.

[0052] Wie im Block **630** gezeigt, prüft die Überwachungsvorrichtung weiter, ob auftretende Buszyklen eine Schreiboperation in die Überwachungsoperation anzeigen oder anzuzeigen scheinen. Wenn ein solcher Buszyklus erkannt wird, wird der Anstehendes-Ereignis-Anzeiger gesetzt, wie im Block **635** angegeben. Nach der Ausführung des MWAIT-Opcodes (Block **505**, **Fig.** 5) wird dieser Anstehendes-Ereignis-Anzeiger als ein Ereignis versorgt und bewirkt Threadwiederaufnahme in den Blöcken **560–580** von **Fig.** 5. Außerdem können Ereignisse, die die Adressenübersetzung verändern, bewirken, daß Thread **1** wiederaufgenommen wird. Zum Beispiel können Ereignisse, die bewirken, daß ein Übersetzungs-Look-Aside-Puffer ausgeräumt wird, die Wiederaufnahme von Thread **1** auslösen, da die Übersetzung, die vorgenommen wurde, um die Überwachungsadresse aus einer linearen zu einer physikalischen Adresse zu erzeugen, möglicherweise nicht mehr gültig ist. Zum Beispiel können in einem mit der x86-Intel-Architektur kompatiblen Prozessor Schreiboperationen in die Steuerregister CR0, CR3 und CR4 sowie in bestimmte maschinenspezifische Register einen Austritt des Speicherwartezustands verursachen.

[0053] Wie bereits erwähnt, zeigt **Fig.** 6b weitere Einzelheiten der Erweiterung der Beobachtbarkeit von Schreiboperationen in die Überwachungsadresse (Block **610**, **Fig.** 6a). Bei einer Ausführungsform räumt der Prozessor die der Überwachungsadresse zugeordnete Cache-Leitung aus allen internen Caches des Prozessors aus, wie im Block **650** angegeben. Als Folge dieses Ausräumens erreicht jede nachfolgende Schreiboperation in die Überwachungsadresse die Busschnittstelle **300**, wodurch eine Erkennung durch die Überwachungsvorrichtung **310**, die in der Busschnittstelle **300** enthalten ist,

möglich wird. Bei einer Ausführungsform wird der MONITOR-uOP nach einer Cache-Leitungs-Ausräum-CLFLUSH-Anweisung, die eine existierende Anweisung in einem x86-Anweisungssatz ist, modelliert bzw. hat dasselbe Fehlermodell. Der Überwachungs-uOP schreitet durch linear-zu-physikalisch-Übersetzung der Adresse und Ausräumen interner Caches sehr ähnlich wie ein CLFLUSH voran; die Busschnittstelle erkennt jedoch den Unterschied zwischen MONITOR und CLFLUSH und behandelt den MONITOR-uOP entsprechend.

[0054] Wie im Block **655** angegeben, aktiviert als nächstes die mit Kohärenz zusammenhängende Logik **350** in der Busschnittstelle **300** die Leseleitungserzeugungslogik **355**, um eine Leseleitungstransaktion auf dem Prozessorbus zu erzeugen. Die Leseleitungstransaktion in die Überwachungsadresse stellt sicher, daß keine anderen Caches in Prozessoren auf dem Bus Daten entweder in einem gemeinsam benutzten oder exklusiven Zustand (entsprechend dem wohlbekannten MESI-Protokoll) an der Überwachungsadresse speichern. Bei anderen Protokollen können andere Zustände verwendet werden; die Transaktion ist jedoch dafür ausgelegt, die Wahrscheinlichkeit zu verringern, daß ein anderer Agent in die Überwachungsadresse schreiben kann, ohne daß die Transaktion durch die Überwachungsvorrichtung **310** beobachtbar ist. Anders ausgedrückt, werden Schreiboperationen oder Schreiben-anzeigende Transaktionen nachfolgend rundgesendet, so daß sie durch die Überwachungsvorrichtung erkannt werden können. Nachdem die Leseleitungsoperation fertig ist, beginnt die Überwachungsvorrichtung **310** mit der Überwachung von Transaktionen auf dem Bus.

[0055] Während zusätzliche Transaktionen auf dem Bus auftreten, bewahrt die mit Kohärenz zusammenhängende Logik weiter die Beobachtbarkeit der Überwachungsadresse, indem versucht wird, zu verhindern, daß Busagenten die Eigentümerschaft der der überwachten Adresse zugeordneten Cache-Leitung übernehmen. Gemäß einem Busprotokoll kann dies dadurch erreicht werden, daß die Treffererzeugungslogik **360** während einer Snoop-Phase jeder Leseoperation der Überwachungsadresse ein HIT#-Signal setzt, wie im Block **660** angegeben. Das Setzen von HIT# verhindert, daß sich andere Caches über den gemeinsam benutzten Zustand hinaus in dem MESI-Protokoll zu dem Exklusiv- und dann potentiell zu dem Modifiziert-Zustand bewegen. Wie im Block **665** angegeben, können folglich keine Agenten in dem gewählten Kohärenzbereich (der Speicherteil, der kohärent gehalten wird) Daten in dem Modifiziert- oder Exklusiv-Zustand (oder ihren Äquivalenten) aufweisen. Der Prozessor scheint effektiv die Cache-Leitung der Überwachungsadresse Cache-gespeichert zu haben, obwohl sie bei dieser Ausführungsform aus internen Caches ausgeräumt wurde.

[0056] Nunmehr mit Bezug auf **Fig.** 7 sind weitere Einzelheiten der dem Block **620** in **Fig.** 6a zugeordneten Operationen aufgeführt. Insbesondere zeigt **Fig.** 7 weitere Einzelheiten der Funktionsweise der Überwachungsvorrichtung **310**. Im Block **700** empfängt die Überwachungsvorrichtung **310** Anforderungs- und Adresseninformationen aus einer Bussteuerung **340** für eine Bustransaktion. Wie im Block **710** angegeben, untersucht die Überwachungsvorrichtung **310** den Buszyklustyp und die betroffene(n) Adresse(n). Insbesondere bestimmt die Zyklusvergleichslogik **320**, ob der Buszyklus ein spezifizierter Zyklus ist. Bei einer Ausführungsform vergleicht eine Adressenvergleichsschaltung **330** die Bustransaktionsadresse mit der in dem Überwachungsadressenregister **335** gespeicherten Überwachungsadresse und Schreibdetektionslogik **325** decodiert die Zyklustypinformationen aus der Bussteuerung **340**, um zu erkennen, ob eine Schreiboperation aufgetreten ist. Wenn eine Schreiboperation in die Überwachungsadresse auftritt, wird ein Anstehendes-Ereignis-Anzeiger gesetzt, wie im Block **720** angegeben. Der Threadsuspendierungs-/-wiederaufnahmelogik **377** wird ein Signal (WRITE DETECTED) zugeführt, um das Ereignis zu signalisieren (und es wird unter der Annahme versorgt, daß sie durch Ausführen von MWAIT freigegeben wurde). Als letztes wird die Überwachungsvorrichtung **310** angehalten, wie im Block **730** angegeben. Das Anhalten der Überwachungsvorrichtung spart Strom, ist aber nicht kritisch, solange falsche Überwachungsereignisse maskiert oder anderweitig nicht erzeugt werden. Auch kann an diesem Punkt der Überwachungsereignisanzeiger zurückgesetzt werden. In der Regel maskiert das Versorgen des Überwachungsereignisses außerdem die Erkennung weiterer Überwachungsereignisse, bis MWAIT nochmals ausgeführt wird.

[0057] Im Fall einer Leseoperation auf die Überwachungsadresse wird die mit Kohärenz zusammenhängende Logik **350** aktiviert. Wie im Block **740** angegeben, wird ein Signal (wie zum Beispiel HIT#) gesetzt, um zu verhindern, daß ein anderer Agent Eigentümerschaft erhält, die zukünftige Schreiboperationen ohne Kohärenzrundsendungen erlauben würde. Die Überwachungsvorrichtung **310** bleibt aktiv und kehrt danach zum Block **700** zurück und wird nicht durch eine Leseoperation der Überwachungsadresse beeinflußt. Wenn eine Transaktion weder eine Leseoperation noch eine Schreiboperation in die Überwachungsadresse ist, bleibt die Überwachungsvorrichtung zusätzlich aktiv und kehrt zum Block **700** zurück.

[0058] Bei bestimmten Ausführungsformen wird die MONITOR-Anweisung so begrenzt, daß nur bestimmte Arten von Zugriffen überwacht werden können. Diese Zugriffe können Zugriffe sein, die als effiziente Programmiertechniken anzeigend gewählt werden, oder können aus anderen Gründen gewählt

werden. Zum Beispiel muß bei einer Ausführungsform der Speicherzugriff eine cachebare Speicheroperation in dem Rückschreibespeicher sein, die naturgemäß ausgerichtet ist. Ein naturgemäß ausgerichtetes Element ist ein N-Bitelement, das an einer durch N teilbaren Adresse startet. Als Folge der Verwendung naturgemäß ausgerichteter Elemente muß auf eine einzige Cache-Leitung zugegriffen werden (statt auf zwei Cache-Leitungen, so wie es notwendig wäre, falls Daten über zwei Cache-Leitungen verteilt werden), um in die überwachte Adresse zu schreiben. Folglich kann die Verwendung naturgemäß ausgerichteter Speicheradressen das Beobachten des Busses vereinfachen.

[0059] **Fig.** 8 zeigt eine Ausführungsform eines Systems, die Mehrfach-Thread-Speicher-Wartetechniken verwendet. Bei der Ausführungsform von **Fig.** 8 ist eine Menge von N Mehrfach-Thread-Prozessoren (Prozessoren **805-1** bis **805-N**) an einen Bus **802** angekoppelt. Bei anderen Ausführungsformen kann ein einziger Prozessor oder eine Mischung von Mehrfach-Thread-Prozessoren und Einzel-Thread-Prozessoren verwendet werden. Zusätzlich können andere bekannte oder anderweitige verfügbare Systemanordnungen verwendet werden. Zum Beispiel können die Prozessoren in einem Punkt-zu-Punkt-Verfahren verbunden werden, und Teile wie zum Beispiel die Speicherschnittstelle können in jeden Prozessor integriert sein.

[0060] Bei der Ausführungsform von **Fig.** 8 ist eine an den Bus angekoppelte Speicherschnittstelle **815** an einen Speicher **830** und eine Medienschnittstelle angekoppelt. Der Speicher **830** enthält ein für Mehrfach-Verarbeitung bereites Betriebssystem **835** und Anweisungen für einen ersten Thread **840** und Anweisungen für einen zweiten Thread **845**. Die Anweisungen **830** enthalten gemäß den offengelegten Techniken, von denen verschiedene Versionen in **Fig.** 9a–9c gezeigt sind, eine Leerlaufschleife.

[0061] Die entsprechende Software zur Durchführung dieser verschiedenen Funktionen kann in einem beliebigen einer Vielfalt maschinenlesbarer Medien bereitgestellt werden. Die Medienschnittstelle **820** liefert eine Schnittstelle zu solcher Software. Die Medienschnittstelle **820** kann eine Schnittstelle zu einem Speichermedium (z.B. einem Plattenlaufwerk, einem optischen Laufwerk, einem Bandlaufwerk, einem flüchtigen Speicher, einem nichtflüchtigen Speicher oder dergleichen) oder zu einem Übertragungsmedium (z.B. einer Netzwerkschnittstelle oder einer anderen digitalen oder analogen Kommunikationsschnittstelle) sein. Die Medienschnittstelle **820** kann Softwareroutinen aus einem Medium (z.B. dem Speichermedium **792** oder dem Übertragungsmedium **795**) lesen. Maschinenlesbare Medien sind Medien, die Informationen zumindest vorübergehend zum Lesen durch eine Maschinenschnittstelle speichern können. Dazu können Signalübertragungen gehören (über Draht, Optik oder Luft als Medium) und/oder physikalische Speichermedien **792**, wie zum Beispiel verschiedene Arten von Laufwerk- und Speichergeräten.

[0062] **Fig.** 9a zeigt eine Leerlaufschleife gemäß einer Ausführungsform. Im Block **905** wird der MONITOR-Befehl mit Adresse **1** als seinem Operanden (der Überwachungsadresse) ausgeführt. Im Block **910** wird der MWAIT-Befehl in demselben Thread ausgeführt. Wie bereits besprochen, bewirkt die MWAIT-Anweisung, daß der Thread suspendiert wird, vorausgesetzt, daß andere Bedingungen ordnungsgemäß erfüllt sind. Wenn im Block **915** ein Break-Ereignis auftritt, geht die Routine zum Block **920** weiter, um zu bestimmen, ob sich der an der Überwachungsadresse gespeicherte Wert geändert hat. Wenn sich der Wert an der Überwachungsadresse geändert hat, dann wird die Ausführung des Threads fortgesetzt, wie im Block **922** angegeben. Wenn sich der Wert nicht geändert hat, dann ist ein falsches Weckereignis aufgetreten. Das Weckereignis ist insofern falsch, als aus MWAIT ausgetreten wurde, ohne daß eine Speicherschreiboperation in die Überwachungsadresse stattgefunden hat. Wenn sich der Wert nicht verändert hat, kehrt die Schleife zum Block **905** zurück, indem die Überwachungsvorrichtung nochmals eingerichtet wird. Durch diese Schleifensoftwareimplementierung wird es möglich, die Überwachungsvorrichtung so auszulegen, daß falsche Weckereignisse zugelassen werden.

[0063] **Fig.** 9b zeigt eine alternative Leerlaufschleife. Die Ausführungsform von **Fig.** 9b fügt eine zusätzliche Prüfung hinzu, um die Wahrscheinlichkeit, daß die MWAIT-Anweisung eine Schreiboperation in die überwachte Speicheradresse nicht erfaßt, weiter zu reduzieren. Wieder beginnt der Fluß in **Fig.** 9b mit der Ausführung der MONITOR-Anweisung mit Adresse **1** als ihrem Operanden, wie im Block **925** angegeben. Zusätzlich liest die Softwareroutine im Block **930** den Speicherwert an der Überwachungsadresse. Im Block **935** führt die Software eine Nachprüfung durch, um sicherzustellen, daß sich der Speicherwert nicht von dem Wert geändert hat, wodurch angezeigt wird, daß der Thread in Leerlauf versetzt werden sollte. Wenn sich der Wert geändert hat, wird die Threadausführung fortgesetzt, wie im Block **952** angegeben. Wenn sich der Wert nicht geändert hat, wird MWAIT-Anweisung ausgeführt, wie im Block **940** angegeben. Wie bereits besprochen, wird der Thread suspendiert, bis ein Break-Ereignis auftritt (Block **945**). Da falsche Break-Ereignisse zugelassen werden, wird jedoch im Block **950** nochmals geprüft, ob sich der Wert geändert hat. Wenn sich der Wert nicht geändert hat, kehrt die Schleife zurück, um nochmals die Überwachungsvorrichtung freizugeben, Adresse **1** zu verfolgen, indem zu Block **925** zurückgekehrt wird. Wenn sich der Wert geändert hat, wird die Aus-

führung des Threads im Block **952** fortgesetzt. Bei bestimmten Ausführungsformen muß die MONITOR-Anweisung nach einem falschen Weckereignis nicht noch mal ausgeführt werden, bevor die MWAIT-Anweisung ausgeführt wird, um den Thread nochmals zu suspendieren.

[0064] **Fig.** 9c zeigt ein weiteres Beispiel für eine Softwaresequenz, die MONITOR- und MWAIT-Anweisungen verwendet. In dem Beispiel von **Fig.** 9c läuft die Schleife nur dann leer, wenn zwei getrennte Tasks in demselben Thread nichts zu tun haben. An der Arbeitsspeicherstelle WL1 wird ein konstanter Wert CV1 gespeichert, wenn eine erste Routine Arbeit zu leisten hat. Ähnlich wird in WL2 ein zweiter konstanter Wert CV2 gespeichert, wenn eine zweite Routine Arbeit zu leisten hat. Um eine einzige Überwachungsadresse zu verwenden, werden WL1 und WL2 als Speicherstellen in derselben Cache-Leitung gewählt. Als Alternative kann auch eine einzige Arbeitsspeicherstelle verwendet werden, um Statusanzeiger für mehrere Tasks zu speichern. Zum Beispiel können ein oder mehrere Bits in einem einzigen Byte oder in einer anderen Einheit jeweils eine verschiedene Tasks repräsentieren.

[0065] Wie im Block **955** angegeben, wird die Überwachungseinrichtung dafür eingerichtet, WL1 zu überwachen. Im Block **960** wird geprüft, ob WL1 den konstanten Wert speichert, wodurch angezeigt wird, daß Arbeit zu verrichten ist. Wenn dies der Fall ist, wird die WL1 betreffende Arbeit durchgeführt, wie im Block **965** angegeben. Wenn nicht, wird im Block **970** geprüft, ob WL2 CV2 speichert, wodurch angezeigt wird, daß Arbeit im Zusammenhang WL2 zu verrichten ist. Wenn dies der Fall ist, wird die mit WL2 zusammenhängende Arbeit verrichtet, wie im Block **975** angegeben. Wenn nicht, kann die Schleife dann bestimmen, ob es angemessen ist, einen Stromverwaltungs-Handler aufzurufen (Block **980**). Wenn zum Beispiel ein gewählter Zeitraum abgelaufen ist, kann der logische Prozessor in einen Zustand mit verringerter Stromaufnahme versetzt werden (z.B. in einen einer Menge „C"-Zuständen, die unter der Advanced Configuration and Power Interface (ACPI) Specification, Version 1.0b (oder später), veröffentlicht am 8.2.1999, erhältlich bei www.acpi.info zum Zeitpunkt der Registration der vorliegenden Anmeldung definiert werden). Wenn dies der Fall ist, wird im Block **985** der Stromverwaltungs-Handler aufgerufen. In jedem der Fälle **965**, **975** und **985**, in denen Arbeit zu verrichten war, verrichtet der Thread diese Arbeit und kehrt dann in einer Schleife zurück, um dieselben Bestimmungen nach der Einrichtung der Überwachungsvorrichtung im Block **955** vorzunehmen. Bei einer alternativen Ausführungsform könnte die Schleife zurück von den Blöcken **965**, **975** und **985** zu dem Block **960** gehen, solange die Überwachungsvorrichtung aktiv bleibt.

[0066] Wenn durch die Blöcke **965**, **975** und **985** hindurch keine zu verrichtende Arbeit angetroffen wird, dann wird die MWAIT-Anweisung ausgeführt, wie im Block **990** angegeben. Der durch MWAIT verursachte Thread-Suspendiert-Zustand wird schließlich verlassen, wenn ein Break-Ereignis auftritt, wie im Block **995** angegeben. An diesem Punkt kehrt die Schleife zum Block **955** zurück, um die Überwachungsvorrichtung einzurichten und um danach zu bestimmen, ob entweder WL1 oder WL2 angeben, daß Arbeit zu verrichten ist. Wenn keine Arbeit zu verrichten ist (z.B. im Fall eines falschen Aufweckereignisses), kehrt die Schleife im Block **990** zu MWAIT zurück und suspendiert den Thread wieder, bis ein Break-Ereignis auftritt.

[0067] **Fig.** 10 zeigt eine alternative Ausführungsform eines Prozessors, die es ermöglicht, daß der Überwachungswert in dem L1-Cache Cache-gespeichert bleibt. Der Prozessor in **Fig.** 10 enthält Ausführungseinheiten **1005**, einen L1-Cache **1010** und Schreibkombinierpuffer zwischen dem L1-Cache und einem inklusiven L2-Cache **1030**. Die Schreibkombinierpuffer **1020** enthalten einen Snoop-Port **1044**, der Kohärenz der internen Caches mit anderem Speicher über durch eine Busschnittstelle **1040** aus einem Bus **1045** empfangene Operationen sicherstellt. Da sich auf Kohärenz auswirkende Transaktionen die Schreibkombinierpuffer **1020** über den Snoop-Port **1044** erreichen, kann sich eine Überwachungsvorrichtung auf der Ebene des L1-Cache befinden und immer noch ausreichend Informationen empfangen, um zu bestimmen, wann ein Speicherschreibereignis auf dem Bus **1045** auftritt. Somit kann die Leitung des Speichers, die der Überwachungsadresse entspricht, in dem L1-Cache gehalten werden. Die Überwachungsvorrichtung kann sowohl Schreiboperationen in den L1-Cache aus den Ausführungseinheiten als auch Schreiboperationen aus dem Bus **1045** über den Snoop-Port **1044** erkennen.

[0068] Eine weitere alternative Ausführungsform unterstützt eine Zweioperandenüberwachungsanweisung. Ein Operand gibt wie zuvor besprochen die Speicheradresse an. Der zweite Operand ist eine Maske, die angibt, welches einer Vielfalt von Ereignissen, die ansonsten kein Break von dem Speicherwartezustand bewirken würden, ein Break von diesem bestimmten Speicherwarten verursachen sollte. Zum Beispiel kann ein Maskenbit angeben, daß maskierte Interrupts zugelassen werden sollten, um ein Break des Speicherwartens zu bewirken, obwohl die Interrupts maskiert sind (z.B. Zulassen eines Aufweckereignisses auch wenn das EFLAGS-Bit IF gesetzt ist, um Interrupts zu maskieren). Es ist anzunehmen, daß dann eine der Anweisungen, die nach dem Break des Speicherwartezustands ausgeführt werden, diesen Interrupt entmaskiert, so daß er versorgt wird. Andere Ereignisse, die ansonsten kein Break des Speicherwartezustands bewirken würden, kön-

nen freigegeben werden, ein Break des Speicherwartens zu bewirken, oder umgekehrt können Ereignisse, die normalerweise ein Break des Speicherwartezustands bewirken, können gestört werden. Wie bei dem Operanden besprochen, kann der zweite Operand explizit oder implizit sein.

[0069] **Fig.** 11 zeigt verschiedene Entwurfsrepräsentationen oder -formate zur Simulation, Emulation und Herstellung eines Entwurfs unter Verwendung der offengelegten Techniken. Daten, die einen Entwurf repräsentieren, können den Entwurf auf vielerlei Weise repräsentieren. Erstens ist es in Simulationen nützlich, daß die Hardware mit einer Hardwarebeschreibungssprache oder einer anderen funktionalen Beschreibungssprache repräsentiert wird, die im wesentlichen ein computerisiertes Modell dafür liefert, wie die entworfene Hardware erwartungsgemäß arbeiten wird. Das Hardwaremodell **1110** kann in einem Speichermedium **1100**, wie zum Beispiel einem Computerspeicher, gespeichert werden, so daß das Modell unter Verwendung von Simulationssoftware **1120** simuliert werden kann, die eine bestimmte Prüfsuite **1130** auf das Hardwaremodell **1110** anwendet, um zu bestimmen, ob es tatsächlich wie beabsichtigt funktioniert. Bei bestimmten Ausführungsformen wird die Simulationssoftware nicht aufgezeichnet, erfaßt oder in dem Medium gehalten.

[0070] Zusätzlich kann ein Modell auf Schaltungsebene mit Logik und/oder Transistorgattern in einer bestimmten Phase des Entwurfsprozesses erzeugt werden. Dieses Modell kann ähnlich simuliert werden, und zwar manchmal durch eigene Hardwaresimulatoren, die das Modell unter Verwendung programmierbarer Logik bilden. Diese Art von Simulation kann, wenn sie etwas weiter geführt wird, eine Emulationstechnik sein. In jedem Fall ist unkonfigurierbare Hardware eine weitere Ausführungsform, die ein maschinenlesbares Medium beteiligen kann, das ein Modell speichert, das die offengelegten Techniken verwendet.

[0071] Außerdem erreichen die meisten Entwürfe in einer bestimmten Phase eine Ebene von Werten, die die physikalische Plazierung verschiedener Geräte in dem Hardwaremodell repräsentieren. Falls herkömmliche Halbleiterherstellungstechniken verwendet werden, können die das Hardwaremodell repräsentierenden Daten die Daten sein, die die Anwesenheit oder Abwesenheit verschiedener Merkmale auf verschiedenen Maskenschichten für zur Herstellung der integrierten Schaltung verwendete Masken spezifizieren. Wiederum realisieren diese die integrierte Schaltung repräsentierenden Daten insofern die offengelegten Techniken, als die Schaltkreise oder Logik in den Daten simuliert oder hergestellt werden, um diese Techniken durchzuführen.

[0072] Bei jeder Repräsentation des Entwurfs können die Daten in jeder beliebigen Form eines computerlesbaren Mediums gespeichert werden. Eine optische oder elektrische Welle **1160**, die moduliert oder anderweitig erzeugt wird, um solche Informationen zu senden, ein Speicher **1150** oder eine magnetische oder optische Speicherung **1140**, zum Beispiel eine Platte, können das Medium sein. Die Menge von Bits, die den Entwurf oder den bestimmten Teil des Entwurfs beschreiben, sind ein Artikel, der an und für sich von anderen zum weiteren Entwurf oder zur weiteren Herstellung verkauft oder benutzt werden kann.

[0073] Es werden also Techniken zum Suspendieren der Ausführung eines Threads bis ein spezifizierter Speicherzugriff auftritt, offengelegt. Obwohl bestimmte Ausführungsbeispiele beschrieben und in den beigefügten Zeichnungen gezeigt wurden, versteht sich, daß solche Ausführungsformen die allgemeine Erfindung lediglich veranschaulichen und nicht einschränken, und daß die vorliegende Erfindung nicht auf die spezifisch gezeigten und beschriebenen Konstruktionen und Anordnungen beschränkt ist, da Durchschnittsfachleuten bei Durchsicht der vorliegenden Offenlegung verschiedene andere Modifikationen einfallen können.

Zusammenfassung

[0074] Techniken zum Suspendieren der Ausführung eines Threads, bis ein spezifizierter Speicherzugriff auftritt. Bei einer Ausführungsform enthält ein Prozessor mehrere Ausführungseinheiten, die mehrere Threads ausführen können. Ein erster Thread enthält eine Anweisung, die eine Überwachungsadresse spezifiziert. Suspendierungslogik suspendiert die Ausführung des ersten Threads und eine Überwachungsvorrichtung bewirkt die Wiederaufnahme des ersten Threads als Reaktion auf einen Zugriff auf die spezifizierte Überwachungsadresse.

Patentansprüche

1. Prozessor, umfassend:
mehrere Ausführungseinheiten, um die Ausführung mehrerer Threads zu ermöglichen, einschließlich eines ersten Threads, wobei der erste Thread eine erste Anweisung mit einem zugeordneten Adressenoperanden, der eine Überwachungsadresse angibt, aufweist;
Suspendierungslogik zum Suspendieren der Ausführung des ersten Threads;
eine Überwachungsvorrichtung zur Bewirkung der Wiederaufnahme des ersten Threads als Reaktion auf einen Speicherzugriff auf die Überwachungsadresse.

2. Prozessor nach Anspruch 1, wobei die Überwachungsvorrichtung die Wiederaufnahme als Reaktion auf den Speicherzugriff nur dann bewirken soll, wenn der Speicherzugriff eine tatsächliche oder po-

tentielle Schreiboperation in die Überwachungsadresse angibt.

3. Prozessor nach Anspruch 1, wobei die Überwachungsvorrichtung die Wiederaufnahme des ersten Threads als Reaktion auf den Speicherzugriff auf die Überwachungsadresse bewirken soll, wenn der erste Thread suspendiert wird und Überwachungsereignisse entmaskiert werden.

4. Prozessor nach Anspruch 3, weiterhin mit Ereignisdetektionslogik, um eine Wiederaufnahme des ersten Threads als Reaktion auf ein von dem Speicherzugriff verschiedenes Ereignis zu bewirken.

5. Prozessor nach Anspruch 4, wobei das Ereignis ein Interrupt ist.

6. Prozessor nach Anspruch 1, wobei der zugeordnete Adressenoperand ein impliziter Operand ist.

7. Prozessor nach Anspruch 6, wobei der zugeordnete Adressenoperand in einem vorbestimmten Register gespeichert wird.

8. Prozessor nach Anspruch 1, wobei die Suspendierungslogik die Ausführung des ersten Threads als Reaktion auf eine zweite Anweisung suspendieren soll, wobei die erste Anweisung die Überwachungsvorrichtung freigibt und die zweite Anweisung durch die Überwachungsvorrichtung signalisierte Ereignisse entmaskiert.

9. Prozessor nach Anspruch 8, wobei die zweite Anweisung die Überwachungsvorrichtung nur dann freigibt, wenn die erste Anweisung ausgeführt wurde.

10. Prozessor nach Anspruch 1, wobei die Suspendierungslogik die Ausführung des ersten Threads als Reaktion auf die erste Anweisung suspendieren soll.

11. Prozessor nach Anspruch 8, weiterhin umfassend: Kohärenzlogik zur Verbesserung der Visibilität von Speicheroperationen in die Überwachungsadresse.

12. Prozessor nach Anspruch 11, wobei die Kohärenzlogik sicherstellen soll, daß kein Cache in einem Kohärenzbereich Informationen an der überwachten Adresse in einem Modifiziert- oder Exklusiv-Zustand speichert.

13. Prozessor nach Anspruch 12, wobei die Kohärenzlogik eine der überwachten Adresse zugeordnete Cache-Leitung aus etwaigen internen Caches ausräumen und eine Busleseleitungstransaktion der der Überwachungsadresse zugeordneten Cache-Leitung zu anderen an den Prozessor angekoppelten Prozessoren erzeugen soll, wobei die Busleseleitungstransaktion eine Mehrphasentransaktion ist, die gemäß einem Pipeline-Busprotokoll bereitgestellt wird.

14. Prozessor nach Anspruch 11, wobei die Kohärenzlogik bewirken soll, daß der Prozessor einen Buszyklus erzeugt, um zu verhindern, daß etwaige andere Busagenten eine Schreibtransaktion zu der Überwachungsadresse durchführen, ohne die Schreibtransaktion rund zu senden.

15. Prozessor nach Anspruch 14, weiterhin mit Bussteuerlogik zum Setzen eines Treffersignals als Reaktion darauf, daß ein anderer Busagent Informationen an der Überwachungsadresse liest.

16. Prozessor nach Anspruch 1, wobei die durch den zugeordneten Adressenoperanden angegebene Überwachungsadresse eine Cache-Leitung, einen Teil einer Cache-Leitung oder eine alternativ bemessene Einheit für Daten an einer durch den zugeordneten Adressenoperanden angegebenen Adresse angibt.

17. Prozessor nach Anspruch 1, weiterhin mit Adressenübersetzungslogik zum Übersetzen des zugeordneten Adressenoperanden in die Überwachungsadresse, die eine physikalische Adresse ist.

18. Prozessor nach Anspruch 1, wobei die Überwachungsadresse aus der folgenden Menge gewählt wird: eine physikalische Adresse, eine virtuelle Adresse, eine relative Adresse und eine lineare Adresse.

19. Prozessor nach Anspruch 1, weiterhin mit mehreren partitionierbaren Betriebsmitteln, die partitioniert werden sollen, um einen Teil jedes partitionierbaren Betriebsmittels jedem aktiven der mehreren Threads fest zuzuordnen, wenn mehrere Threads aktiv sind, wobei die Suspendierungslogik etwaige der mehreren dem ersten Thread fest zugeordneten Partitionen als Reaktion auf das Suspendieren der Ausführung des ersten Threads freigeben soll.

20. Prozessor nach Anspruch 19, wobei die Überwachungsvorrichtung bewirken soll, daß die mehreren partitionierbaren Betriebsmittel umpartitioniert werden, um die Ausführung des ersten Threads als Reaktion auf den Speicherzugriff auf die Überwachungsadresse zu ermöglichen.

21. Prozessor nach Anspruch 20, wobei die mehreren partitionierbaren Betriebsmittel folgendes umfassen:
eine Anweisungsschlange;
einen Umordnungspuffer;
ein Registerpool;
mehrere Speicherpuffer.

22. Prozessor nach Anspruch 21, weiterhin umfassend;
mehrere duplizierte Betriebsmittel, wobei die mehreren duplizierten Betriebsmittel für jeden der mehreren Threads dupliziert werden, wobei die mehreren duplizierten Betriebsmittel folgendes umfassen:
mehrere Prozessorzustandsvariablen;
einen Anweisungszeiger;
Registerumbenennungslogik.

23. Prozessor nach Anspruch 22, weiterhin umfassend:
mehrere gemeinsam benutzte Betriebsmittel, wobei die mehreren gemeinsam benutzten Betriebsmittel zur Verwendung durch beliebige der mehreren Threads verfügbar sind, wobei die mehreren gemeinsam benutzten Betriebsmittel folgendes umfassen:
die mehreren Ausführungseinheiten;
einen Cache;
einen Scheduler.

24. Prozessor, umfassend:
ein Frontend zum Empfangen einer ersten Anweisung aus einem ersten Thread und einer zweiten Anweisung aus dem ersten Thread, wobei die erste Anweisung eine Überwachungsadresse angibt;
Ausführungsbetriebsmittel zum Ausführen der ersten Anweisung und der zweiten Anweisung und zum Suspendieren der Ausführung des ersten Threads als Reaktion auf die zweite Anweisung;
eine Überwachungsvorrichtung zur Bewirkung der Wiederaufnahme des ersten Threads als Reaktion auf einen Speicherzugriff auf die Überwachungsadresse.

25. Prozessor nach Anspruch 24, wobei die erste Anweisung einen Operanden aufweist, der eine lineare Adresse angibt, und wobei der Prozessor weiterhin Adresseübersetzungslogik zum Übersetzen der linearen Adresse umfaßt, um die Überwachungsadresse zu erhalten, die eine physikalische Adresse ist.

26. Prozessor nach Anspruch 25, weiterhin umfassend: Kohärenzlogik, um sicherzustellen, daß kein Cache in einem anderen Prozessor, der an den Prozessor angekoppelt ist, Informationen an der Überwachungsadresse in einem Modifiziert- oder Exklusiv-Zustand speichert.

27. Prozessor nach Anspruch 26, wobei die Kohärenzlogik als Reaktion auf ein Snooping der Überwachungsadresse durch einen anderen Prozessor ein Treffersignal setzen soll.

28. Prozessor, umfassend:
Frontendlogik zum Empfangen einer ersten Anweisung aus einem ersten Thread, wobei die erste Anweisung eine zugeordnete Überwachungsadresse aufweist;
eine Überwachungsvorrichtung, die so gekoppelt ist, daß sie die Überwachungsadresse empfängt und als Reaktion auf die erste Anweisung Speicherzugriffe auf die Überwachungsadresse überwacht und ein Ereignis signalisiert, wenn eine Zugriff auf die Überwachungsadresse auftritt.

29. Prozessor nach Anspruch 28, wobei die Überwachungsvorrichtung das Ereignis als Reaktion auf einen Schreibspeicherzugriff, der in die Überwachungsadresse schreibt, signalisieren soll.

30. Prozessor nach Anspruch 28, wobei die Überwachungsvorrichtung das Ereignis als Reaktion auf eine Leitungsinvalidierungstransaktion signalisieren soll.

31. Prozessor nach Anspruch 28, weiterhin umfassend: Kohärenzlogik, um sicherzustellen, daß kein Cache in einem anderen Prozessor, der an den Prozessor angekoppelt ist, Informationen an der Überwachungsadresse in einem Modifiziert- oder Exklusiv-Zustand speichert.

32. Prozessor nach Anspruch 31, wobei die Kohärenzlogik Logik zu Erzeugung eines internen Cache-Ausräumzyklus und zur Erzeugung einer externen Leseleitungstransaktion umfaßt.

33. Prozessor nach Anspruch 28, weiterhin umfassend: Logik zum Entmaskieren von Überwachungsereignissen aus der Überwachungsvorrichtung und zum Suspendieren des ersten Threads als Reaktion auf eine zweite Anweisung.

34. Prozessor, umfassend:
mehrere Ausführungseinheiten zur Ausführung mehrerer Threads;
Frontendlogik zum Empfangen einer Anweisung aus einem ersten Thread der mehreren Threads;
Suspendierungslogik zum Suspendieren des ersten Threads als Reaktion auf die Anweisung, wenn keine Überwachungsereignisse anstehen, und zum Zulassen der Ausführung anderer der mehreren Threads.

35. Prozessor nach Anspruch 34, wobei die Suspendierungslogik die Erkennung von Überwachungsereignissen, einschließlich bereits anstehender Überwachungsereignisse, freigeben soll.

36. Prozessor nach Anspruch 35, wobei der Prozessor mehrere partitionierbare Betriebsmittel umfaßt und wobei die Suspendierungslogik Partitionen jedes der mehreren partitionierbaren Betriebsmittel, die dem ersten Thread zugeordnet sind, zusätzlich zu den Suspendierem des ersten Threads als Reaktion auf die Anweisung freigeben soll.

37. Prozessor, umfassend:
mehrere Thread-partitionierbare Betriebsmittel zum

Empfangen von Anweisungen;
mehrere gemeinsam benutzte Betriebsmittel zur Ausführung von Anweisungen in Zusammenarbeit mit den mehreren Thread-partitionierbaren Betriebsmitteln;
Thread-Suspendierungslogik zum Suspendieren eines ersten Threads als Reaktion auf eine Anweisung in dem ersten Thread, wobei die Thread-Suspendierungslogik Partitionen der mehreren Thread-partitionierbaren Betriebsmittel, die dem ersten Thread zugeordnet sind, als Reaktion auf das Suspendieren des ersten Threads freigeben soll;
eine Überwachungsvorrichtung zum Bewirken, daß als Reaktion auf einen Zugriff auf eine durch den ersten Thread angegebene Speicheradresse der Prozessor die mehreren Thread-partitionierbaren Betriebsmittel neu partitioniert und die Ausführung des ersten Threads wiederaufnimmt.

38. Prozessor nach Anspruch 37, wobei der Zugriff auf die Speicheradresse durch eine erste Anweisung spezifiziert wird, die in dem ersten Thread ausgeführt wird, und wobei die Überwachungsvorrichtung entmaskiert wird, um Überwachungsereignisse zu signalisieren, um eine Thread-Wiederaufnahme durch die Anweisung zu bewirken, woraufhin die Thread-Suspendierungslogik den ersten Thread suspendieren soll.

39. Vorrichtung, umfassend:
Mittel zum Suspendieren eines ersten Threads von mehreren Ausführungsthreads;
Mittel zum Erkennen eines Zugriffs auf eine Speicherstelle;
Mittel zum Wiederaufnehmen des ersten Threads als Reaktion darauf, daß das Mittel zum Erkennen den Zugriff auf die Speicherstelle erkennt.

40. Vorrichtung nach Anspruch 39, wobei das Mittel zum Erkennen des Zugriffs auf die Speicherstelle als Reaktion auf eine in dem ersten Thread ausgeführte erste Anweisung freigegeben wird und wobei das Mittel zum Suspendieren des ersten Threads den ersten Thread als Reaktion auf eine in dem ersten Thread ausgeführte zweite Anweisung suspendiert.

41. Vorrichtung nach Anspruch 40, weiterhin umfassend: ein Kohärenzmittel zum Vereinfachen der Erkennung des Zugriffs auf die Speicherstelle.

42. Vorrichtung nach Anspruch 41, wobei der Zugriff auf die Speicherstelle ein Schreib- oder ein Invalidierungszugriff ist.

43. Vorrichtung nach Anspruch 41, weiterhin umfassend:
ein Mittel zum Verschmelzen von Betriebsmitteln als Reaktion darauf, daß das Mittel zum Suspendieren die Ausführung des ersten Threads suspendiert, wobei das Mittel zum Verschmelzen dem ersten Thread zugeordnete partitionierte Betriebsmittel zur Verwendung durch andere der mehreren Threads befreit;
ein Mittel zum Partitionieren von Betriebsmitteln zum Umpartitionieren von Betriebsmitteln, um die Wiederaufnahme des ersten Threads zu ermöglichen.

44. Verfahren mit den folgenden Schritten:
Empfangen eines ersten Opcodes in einem ersten Ausführungsthread, wobei der erste Opcode einen zugeordneten Adressenoperanden aufweist, der eine Überwachungsadresse angibt;
Suspendieren des ersten Threads;
Erkennen eines Speicherzugriffs auf die Überwachungsadresse;
Wiederaufnehmen des ersten Threads als Reaktion auf das Erkennen des Speicherzugriffs auf die Überwachungsadresse.

45. Verfahren nach Anspruch 44, wobei das Suspendieren des ersten Threads die folgenden Schritte umfaßt:
Empfangen einer zweiten Anweisung in dem ersten Thread;
Suspendieren des ersten Threads als Reaktion auf die zweite Anweisung.

46. Verfahren nach Anspruch 45, wobei der Speicherzugriff ein Schreibzugriff ist.

47. Verfahren nach Anspruch 45, weiterhin mit dem Schritt des Übersetzens des zugeordneten Adressenoperandens in eine überwachte physikalische Adresse, wobei das Erkennen des Speicherzugriffs auf die Überwachungsadresse das Erkennen eines Schreibzugriffs aus die überwachte physikalische Adresse umfaßt.

48. Verfahren nach Anspruch 44, weiterhin mit dem folgenden Schritt: Verhindern, daß andere Agenten die Eigentümerschaft von an der Überwachungsadresse gespeicherten Informationen erhalten.

49. Verfahren nach Anspruch 44, wobei das Erkennen die folgende Schritte umfaßt:
Empfangen von Zyklusinformationen von externen Bustransaktionen;
Erkennen von Schreiboperationen in die Überwachungsadresse

50. Verfahren nach Anspruch 44, weiterhin mit dem folgenden Schritt: Wiederaufnehmen des ersten Threads als Reaktion auf ein von dem Speicherzugriff auf die Überwachungsadresse verschiedenes Ereignis.

51. Verfahren nach Anspruch 50, wobei das Ereignis ein Interrupt ist.

52. Verfahren nach Anspruch 51, wobei der Interrupt ein maskierter Interrupt ist, der durch einen zweiten Operanden angegeben wird, um dennoch als ein Break-Ereignis betrachtet zu werden.

53. Verfahren mit den folgenden Schritten:
Empfangen eines ersten Opcodes, der in einem ersten Ausführungsthread ausgeführt wird;
Übersetzen einer dem ersten Opcode zugeordneten linearen Adresse in eine physikalische Adresse;
Ausführen einer Bustransaktion durch einen überwachenden Busagenten, um sicherzustellen, daß kein anderer Busagent ausreichende Eigentümerschaft von der physikalischen Adresse zugeordneten Daten besitzt, um es einem anderen Busagenten zu erlauben, Daten zu modifizieren, ohne den überwachenden Busagenten zu informieren;
Überwachen auf einen Schreibzugriff auf die physikalische Adresse;
Signalisieren eines Treffers, wenn ein anderer Busagent die physikalische Adresse liest;
Empfangen eines zweiten Opcodes in dem ersten Ausführungsthreads;
Suspendieren des ersten Ausführungsthreads und Freigeben des Erkennens eines Überwachungsereignisses als Reaktion auf den zweiten Opcode;
Wiederaufnehmen des ersten Threads, wenn ein Schreibzugriff auftritt;
Wiederaufnehmen des Ausführens des ersten Threads als Reaktion auf ein beliebiges einer ersten Menge von Ereignissen;
Ignorieren einer zweiten Menge von Ereignissen.

54. Verfahren nach Anspruch 53, wobei das Suspendieren des ersten Ausführungsthreads als Reaktion auf den zweiten Opcode die folgenden Schritte umfaßt:
Prüfen, ob das Überwachungsereignis ansteht;
Prüfen, ob eine Überwachungsvorrichtung aktiv ist;
wenn die Überwachungsvorrichtung aktiv ist und kein Überwachungsereignis ansteht, Eintreten in einen Erster-Thread-Suspendiert-Zustand.

55. Verfahren nach Anspruch 54, wobei der Eintritt in den Erster-Thread-Suspendiert-Zustand die folgenden Schritte umfaßt:
Freigeben mehrerer Register in einem Registerpool;
Freigeben mehrerer Anweisungswarteschlangeneinträge in einer Anweisungswarteschlange;
Freigeben mehrerer Speicherpuffereinträge in einem Speicherpuffer;
Freigeben mehrerer Umordnungspuffereinträge in einem Umordnungspuffer.

56. System, umfassend;
einen Speicher zum Speichern einer ersten Anweisung auf einem ersten Thread, wobei die erste Anweisung einen zugeordneten Adressenoperanden aufweist, der eine Überwachungsadresse angibt;
einen an dem Speicher angekoppelten ersten Prozessor, wobei der erste Prozessor einer Überwachungsvorrichtung ermöglichen soll, Speichertransaktionen zu überwachen, um als Reaktion auf die erste Anweisung einen Speicherzugriff auf die Überwachungsadresse zu erkennen und als Reaktion auf den Speicherzugriff auf die Überwachungsadresse die Wiederaufnahme des ersten Threads zu bewirken.

57. System nach Anspruch 56, wobei der Speicher eine zweite Anweisung aus dem ersten Thread speichern soll und wobei der erste Prozessor den ersten Thread als Reaktion auf die zweite Anweisung suspendieren soll.

58. System nach Anspruch 57, wobei die Überwachungsvorrichtung als Reaktion auf das Auftreten des Speicherzugriffs einen Anstehendes-Überwachungsereignis-Anzeiger setzen soll, wobei der Anstehendes-Überwachungsereignis-Anzeiger bewirken soll, daß der erste Prozessor einen Thread wiederaufnimmt, nachdem er durch die zweite Anweisung entmaskiert wurde.

59. System nach Anspruch 56, wobei der erste Prozessor einen ersten Cache enthält, wobei das System weiterhin folgendes umfaßt: einen zweiten Prozessor mit einem zweiten Cache, wobei der erste Prozessor eine Bustransaktion zu dem zweiten Prozessor steuert, um den zweiten Prozessor dazu zu zwingen, etwaige Transaktionen, die eine Veränderung von an der Überwachungsadresse in dem zweiten Cache gespeicherten Daten ermöglichen, zu dem ersten Prozessor rund zu senden.

60. System nach Anspruch 59, wobei der erste Prozessor ein Signal setzen soll, das verhindert, daß der zweite Prozessor Daten an der Überwachungsadresse in einen Zustand Cache-speichert, der es dem zweiten Prozessor ermöglichen würde, an der Überwachungsadresse in dem zweiten Cache gespeicherte Daten zu modifizieren, ohne rund zu senden, daß eine Modifikation stattfindet.

61. System nach Anspruch 60, wobei das Signal einen Cache-Treffer angibt und verhindert, daß der zweite Cache Daten an der Überwachungsadresse in einen Exklusiv-Zustand speichert.

62. System nach Anspruch 58, wobei der erste Prozessor weiterhin den ersten Thread wiederaufnehmen soll, wenn ein alternatives Ereignis auftritt.

63. System nach Anspruch 62, wobei das alternative Ereignis ein Interrupt ist.

64. System nach Anspruch 62, wobei der in dem Speicher gespeicherte erste Thread eine Schleife enthält, wobei die Schleife die erste Anweisung und die zweite Anweisung sowie eine Prüfung enthält, um

zu bestimmen, ob sich Daten an der Überwachungsadresse verändert haben, und um die Schleife neu zu starten, wenn Daten an der Überwachungsadresse unverändert bleiben.

65. Artikel mit einem computerlesbaren Medium, das einen Prozessor repräsentiert, umfassend:
mehrere Ausführungseinheiten, um die Ausführung mehrerer Threads zu ermöglichen, einschließlich eines ersten Threads, wobei der erste Thread eine erste Anweisung mit einem zugeordneten Adressenoperanden, der eine Überwachungsadresse angibt, aufweist;
Suspendierungslogik zum Suspendieren der Ausführung des ersten Threads; eine Überwachungsvorrichtung zur Bewirkung der Wiederaufnahme des ersten Threads als Reaktion auf einen Speicherzugriff auf die Überwachungsadresse.

66. Artikel nach Anspruch 65, wobei die Überwachungsvorrichtung die Wiederaufnahme als Reaktion auf den Speicherzugriff nur dann bewirken soll, wenn der Speicherzugriff eine tatsächliche oder potentielle Schreiboperation in die Überwachungsadresse angibt.

67. Artikel nach Anspruch 65, wobei die Überwachungsvorrichtung die Wiederaufnahme des ersten Threads als Reaktion auf den Speicherzugriff auf die Überwachungsadresse bewirken soll, wenn der erste Thread suspendiert wird und Überwachungsereignisse entmaskiert werden.

68. Artikel nach Anspruch 65, wobei der Prozessor weiterhin Ereignisdetektionslogik umfaßt, um die Wiederaufnahme des ersten Threads als Reaktion auf ein von dem Speicherzugriff verschiedenes Ereignis zu bewirken.

69. Artikel nach Anspruch 68, wobei der Prozessor weiterhin mehrere partitionierbare Betriebsmittel umfaßt, die partitioniert werden sollen, um einen Teil jedes partitionierbaren Betriebsmittels jedem aktiven der mehreren Threads fest zuzuordnen, wenn mehrere Threads aktiv sind, wobei die Suspendierungslogik alle dem ersten Thread fest zugeordneten Partitionen als Reaktion auf das Suspendieren der Ausführung des ersten Threads freigeben soll.

Es folgen 13 Blatt Zeichnungen

Anhängende Zeichnungen

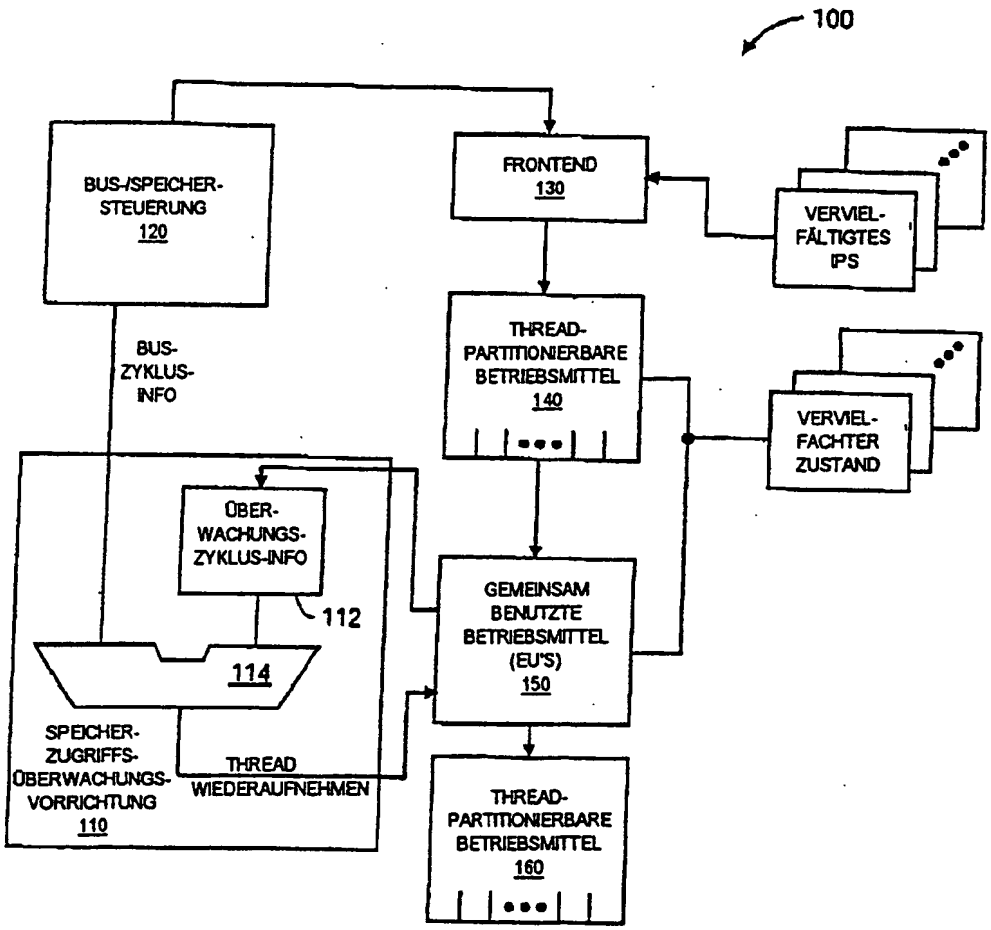


FIG. 1

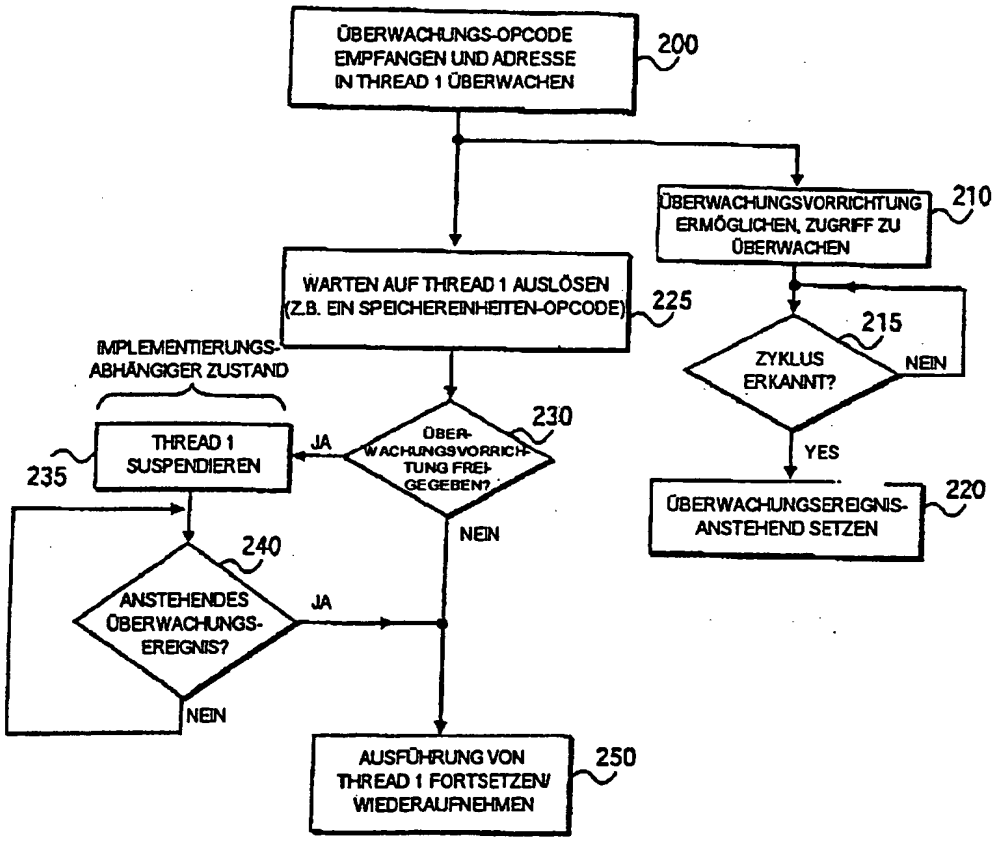

ÜBERWACHUNGS-OPCODE EMPFANGEN UND ADRESSE IN THREAD 1 ÜBERWACHEN
200
ÜBERWACHUNGSVORRICHTUNG ERMÖGLICHEN, ZUGRIFF ZU ÜBERWACHEN
210
WARTEN AUF THREAD 1 AUSLÖSEN (Z.B. EIN SPEICHEREINHEITEN-OPCODE)
225
215
ZYKLUS ERKANNT?
NEIN
YES
IMPLEMENTIERUNGS-ABHÄNGIGER ZUSTAND
ÜBER-WACHUNGSVORRICH-TUNG FREI-GEGEBEN?
230
JA
THREAD 1 SUSPENDIEREN
235
ÜBERWACHUNGSEREIGNIS-ANSTEHEND SETZEN
220
NEIN
240
ANSTEHENDES ÜBERWACHUNGS-EREIGNIS?
JA
NEIN
AUSFÜHRUNG VON THREAD 1 FORTSETZEN/ WIEDERAUFNEHMEN
250

FIG. 2

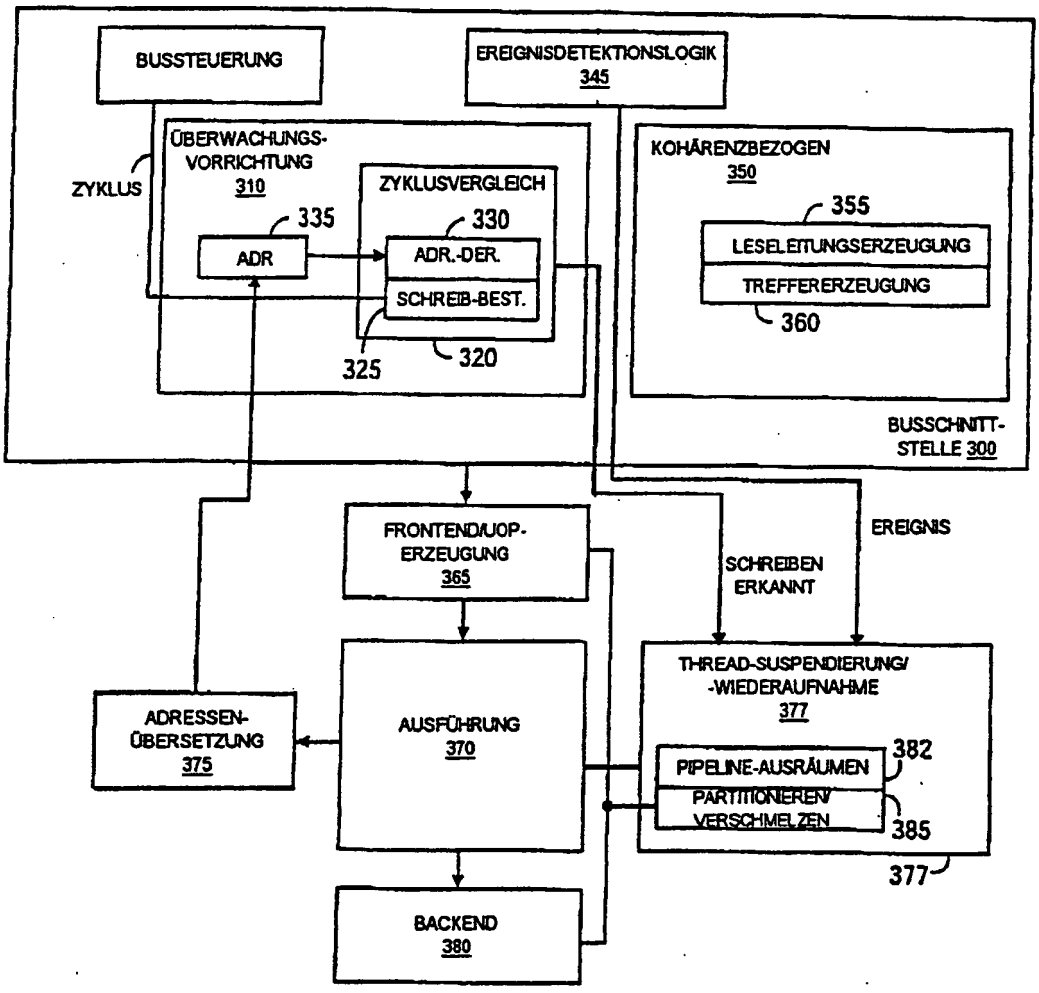

BUSSTEUERUNG
EREIGNISDETEKTIONSLOGIK
345
ÜBERWACHUNGS-
VORRICHTUNG
310
ZYKLUS
ZYKLUSVERGLEICH
335
330
ADR
ADR.-DER.
SCHREIB-BEST.
325
320
KOHÄRENZBEZOGEN
350
355
LESELEITUNGSERZEUGUNG
TREFFERERZEUGUNG
360
BUSSCHNITT-
STELLE 300
FRONTEND/UOP-
ERZEUGUNG
365
SCHREIBEN
ERKANNT
EREIGNIS
THREAD-SUSPENDIERUNG/
-WIEDERAUFNAHME
377
ADRESSEN-
ÜBERSETZUNG
375
AUSFÜHRUNG
370
PIPELINE-AUSRÄUMEN
382
PARTITIONIEREN/
VERSCHMELZEN
385
377
BACKEND
380

FIG. 3

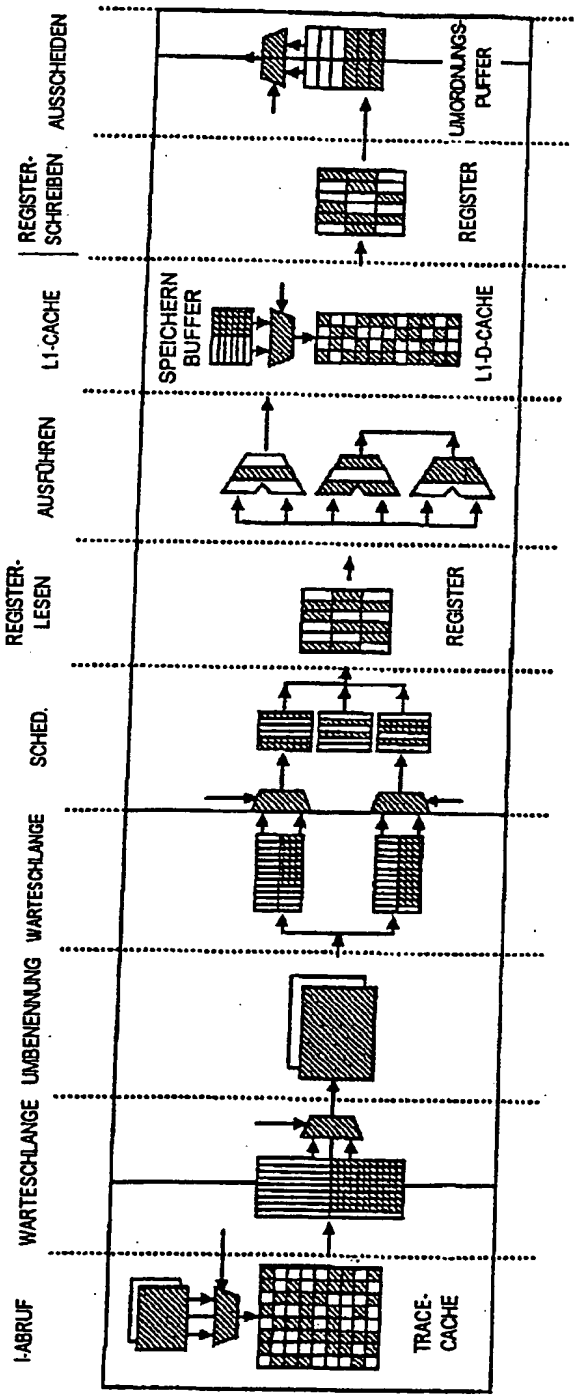

I-ABRUF
WARTESCHLANGE
UMBENENNUNG
WARTESCHLANGE
SCHED.
REGISTER-LESEN
AUSFÜHREN
L1-CACHE
REGISTER-SCHREIBEN
AUSSCHEIDEN
TRACE-CACHE
SPEICHERN BUFFER
L1-D-CACHE
REGISTER
REGISTER
UMORDNUNGS-PUFFER

FIG. 4

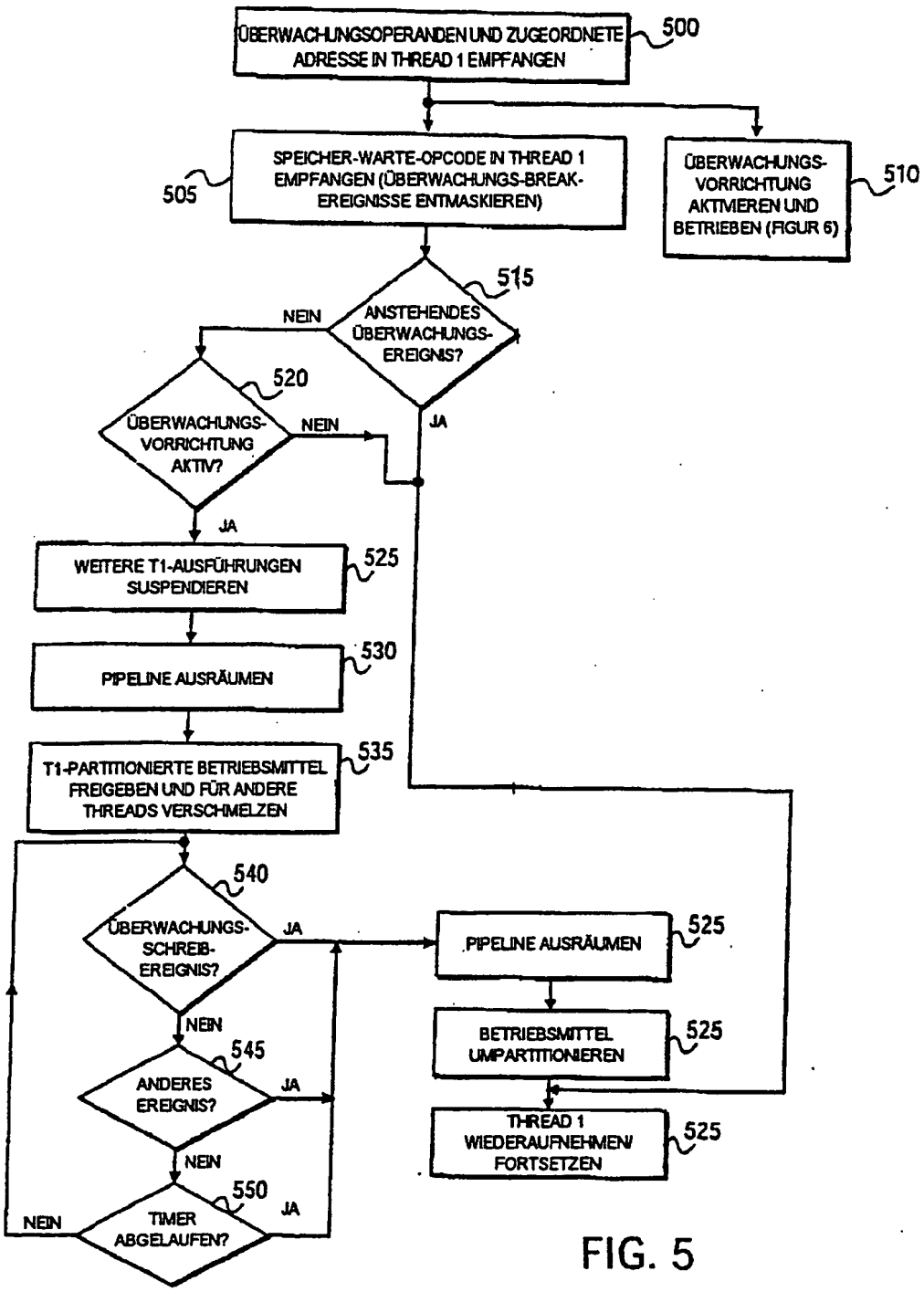

ÜBERWACHUNGSOPERANDEN UND ZUGEORDNETE ADRESSE IN THREAD 1 EMPFANGEN
500
SPEICHER-WARTE-OPCODE IN THREAD 1 EMPFANGEN (ÜBERWACHUNGS-BREAK-EREIGNISSE ENTMASKIEREN)
505
ÜBERWACHUNGS-VORRICHTUNG AKTIVIEREN UND BETREIBEN (FIGUR 6)
510
ANSTEHENDES ÜBERWACHUNGS-EREIGNIS?
515
NEIN
JA
ÜBERWACHUNGS-VORRICHTUNG AKTIV?
520
NEIN
JA
WEITERE T1-AUSFÜHRUNGEN SUSPENDIEREN
525
PIPELINE AUSRÄUMEN
530
T1-PARTITIONIERTE BETRIEBSMITTEL FREIGEBEN UND FÜR ANDERE THREADS VERSCHMELZEN
535
ÜBERWACHUNGS-SCHREIB-EREIGNIS?
540
JA
NEIN
ANDERES EREIGNIS?
545
JA
NEIN
TIMER ABGELAUFEN?
550
JA
NEIN
PIPELINE AUSRÄUMEN
525
BETRIEBSMITTEL UMPARTITIONIEREN
525
THREAD 1 WIEDERAUFNEHMEN/ FORTSETZEN
525

FIG. 5

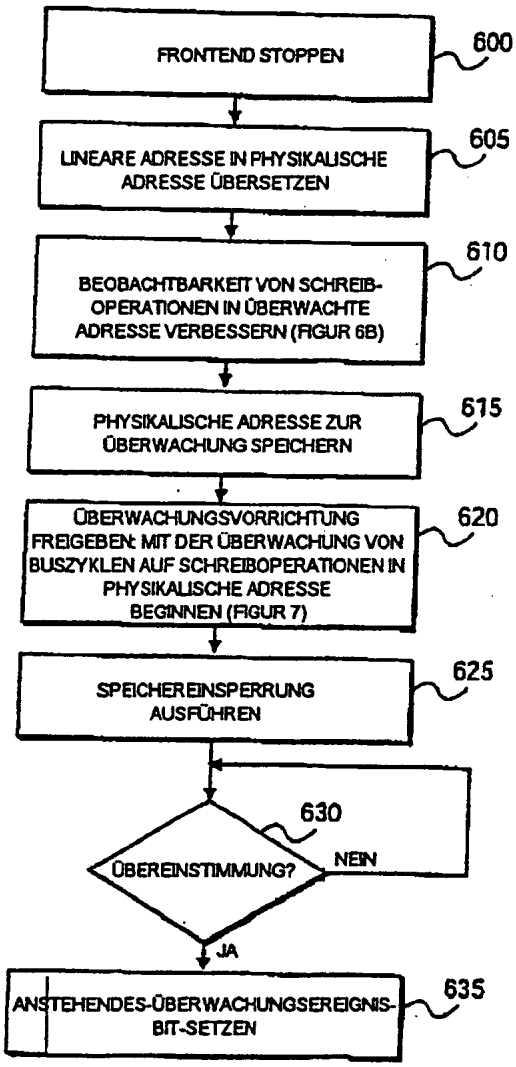

FRONTEND STOPPEN
600
LINEARE ADRESSE IN PHYSIKALISCHE ADRESSE ÜBERSETZEN
605
BEOBACHTBARKEIT VON SCHREIB-OPERATIONEN IN ÜBERWACHTE ADRESSE VERBESSERN (FIGUR 6B)
610
PHYSIKALISCHE ADRESSE ZUR ÜBERWACHUNG SPEICHERN
615
ÜBERWACHUNGSVORRICHTUNG FREIGEBEN: MIT DER ÜBERWACHUNG VON BUSZYKLEN AUF SCHREIBOPERATIONEN IN PHYSIKALISCHE ADRESSE BEGINNEN (FIGUR 7)
620
SPEICHEREINSPERRUNG AUSFÜHREN
625
630
ÜBEREINSTIMMUNG?
NEIN
JA
ANSTEHENDES-ÜBERWACHUNGSEREIGNIS-BIT-SETZEN
635

FIG. 6A

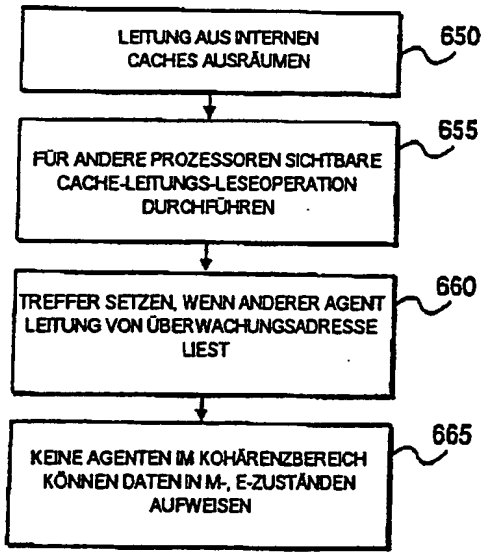

LEITUNG AUS INTERNEN CACHES AUSRÄUMEN
650
FÜR ANDERE PROZESSOREN SICHTBARE CACHE-LEITUNGS-LESEOPERATION DURCHFÜHREN
655
TREFFER SETZEN, WENN ANDERER AGENT LEITUNG VON ÜBERWACHUNGSADRESSE LIEST
660
KEINE AGENTEN IM KOHÄRENZBEREICH KÖNNEN DATEN IN M-, E-ZUSTÄNDEN AUFWEISEN
665

FIG. 6B

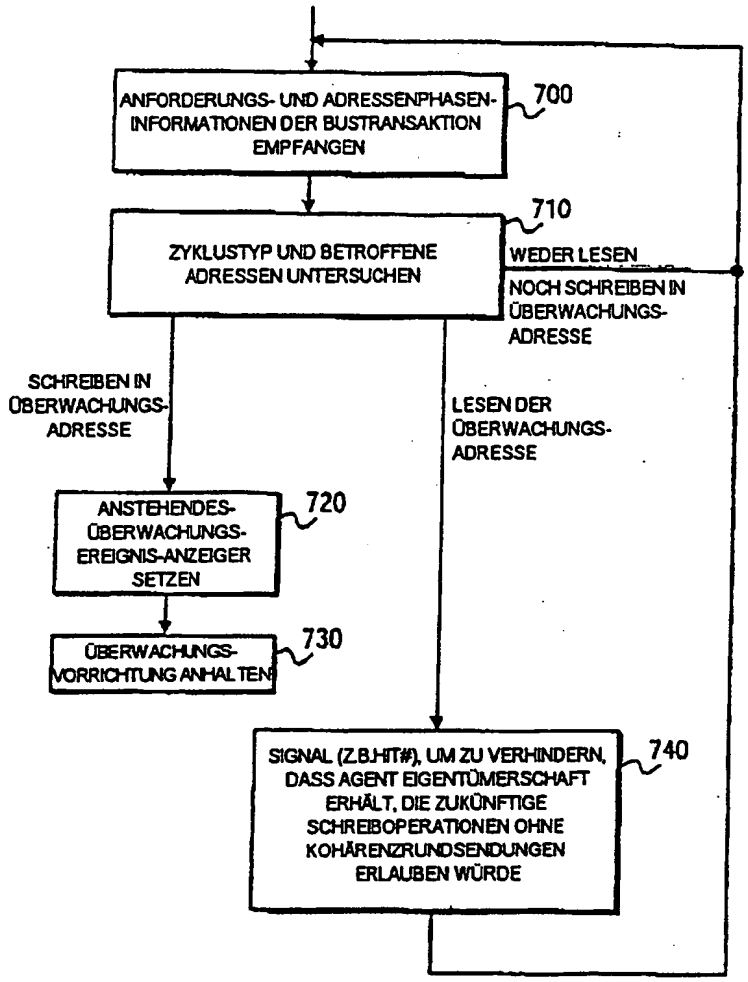

ANFORDERUNGS- UND ADRESSENPHASEN-INFORMATIONEN DER BUSTRANSAKTION EMPFANGEN
700
ZYKLUSTYP UND BETROFFENE ADRESSEN UNTERSUCHEN
710
WEDER LESEN
NOCH SCHREIBEN IN ÜBERWACHUNGS-ADRESSE
SCHREIBEN IN ÜBERWACHUNGS-ADRESSE
LESEN DER ÜBERWACHUNGS-ADRESSE
ANSTEHENDES-ÜBERWACHUNGS-EREIGNIS-ANZEIGER SETZEN
720
ÜBERWACHUNGS-VORRICHTUNG ANHALTEN
730
SIGNAL (Z.B. HIT#), UM ZU VERHINDERN, DASS AGENT EIGENTÜMERSCHAFT ERHÄLT, DIE ZUKÜNFTIGE SCHREIBOPERATIONEN OHNE KOHÄRENZRUNDSENDUNGEN ERLAUBEN WÜRDE
740

FIG. 7

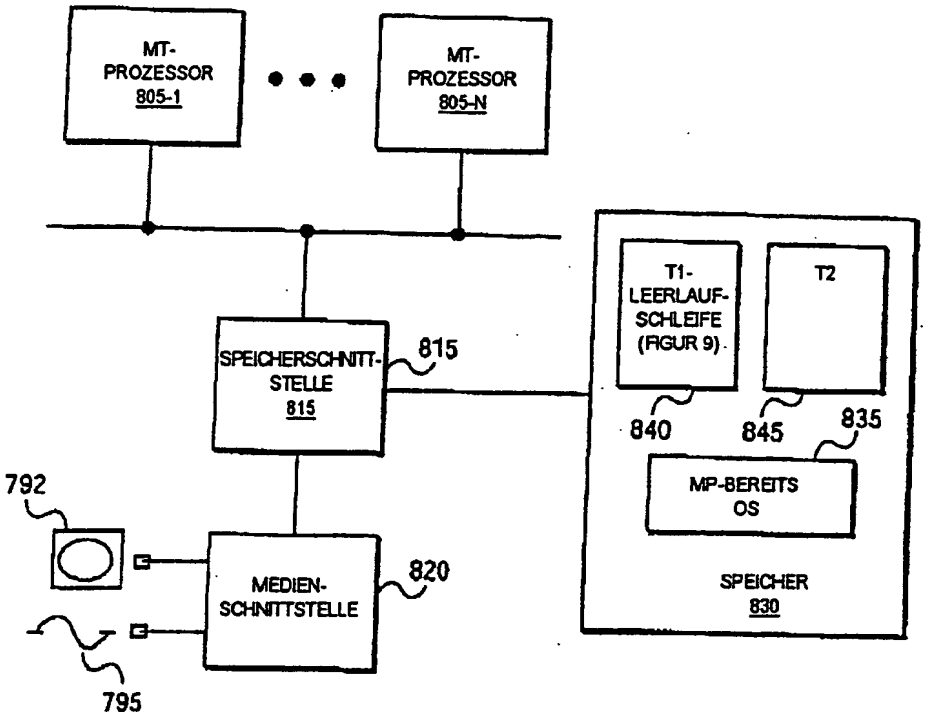

MT-
PROZESSOR
805-1
MT-
PROZESSOR
805-N
SPEICHERSCHNITT-
STELLE
815
815
T1-
LEERLAUF-
SCHLEIFE
(FIGUR 9)
T2
840
845
835
MP-BEREITS
OS
SPEICHER
830
792
MEDIEN-
SCHNITTSTELLE
820
795

FIG. 8

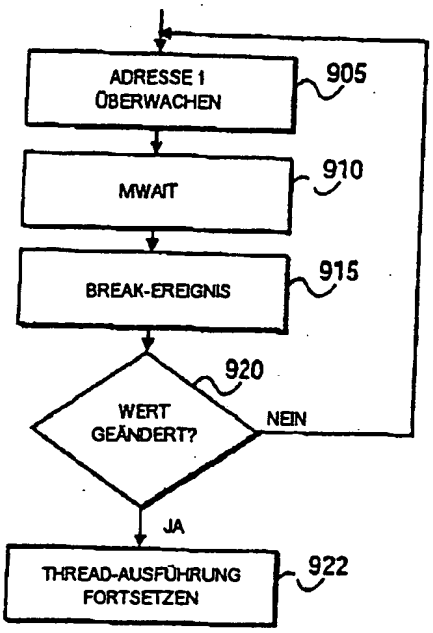


FIG. 9A

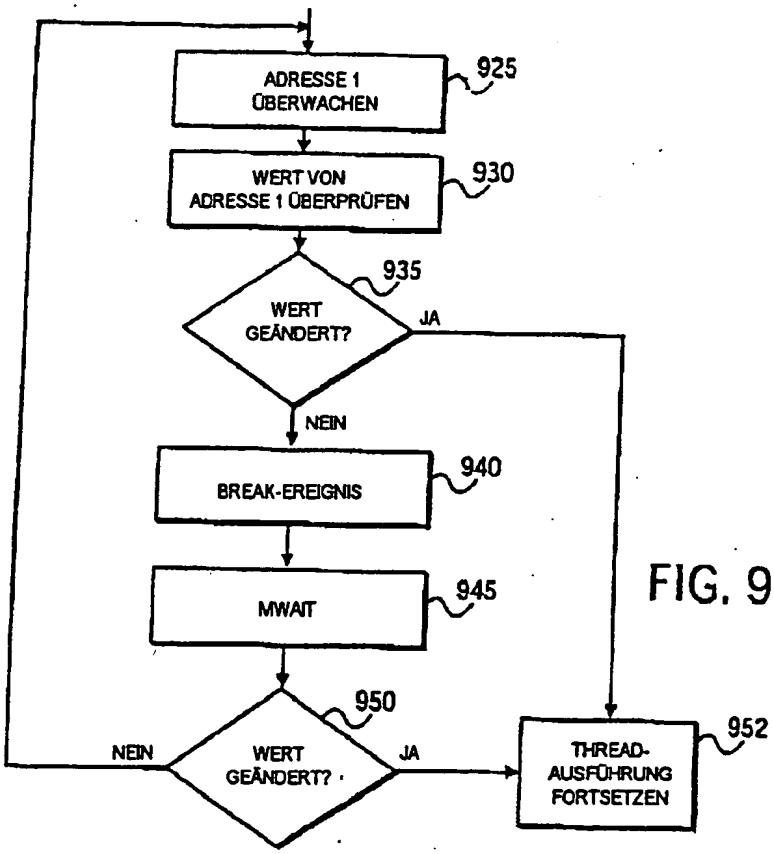


FIG. 9B

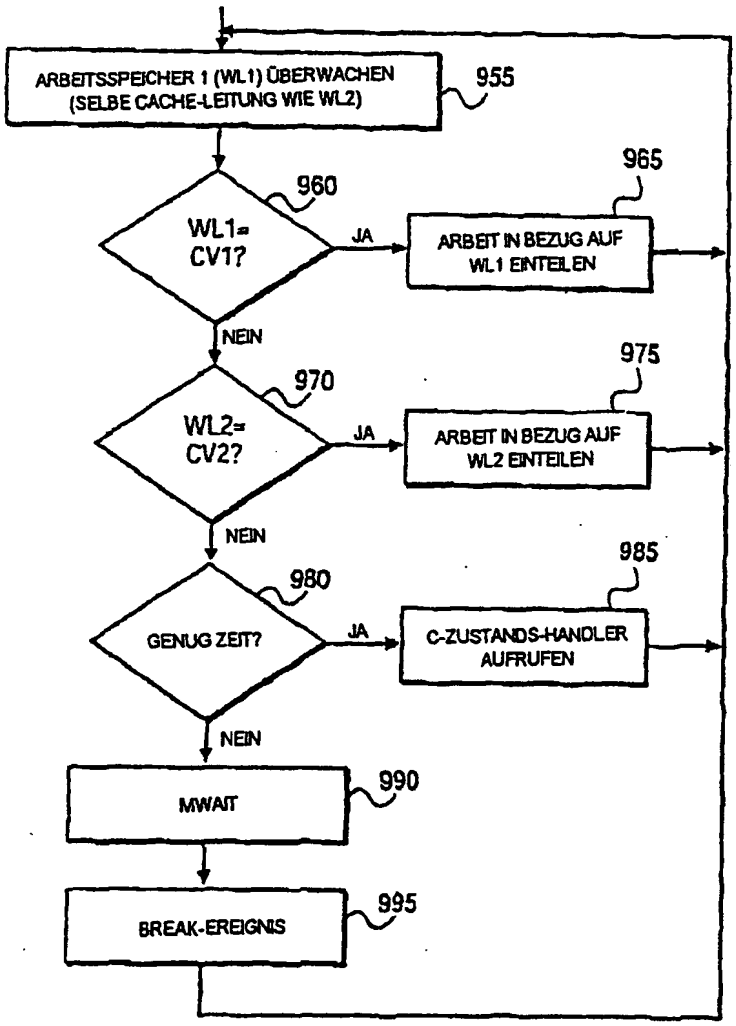

ARBEITSSPEICHER 1 (WL1) ÜBERWACHEN (SELBE CACHE-LEITUNG WIE WL2)
955
960
WL1= CV1?
JA
965
ARBEIT IN BEZUG AUF WL1 EINTEILEN
NEIN
970
WL2= CV2?
JA
975
ARBEIT IN BEZUG AUF WL2 EINTEILEN
NEIN
980
GENUG ZEIT?
JA
985
C-ZUSTANDS-HANDLER AUFRUFEN
NEIN
MWAIT
990
BREAK-EREIGNIS
995

FIG. 9C

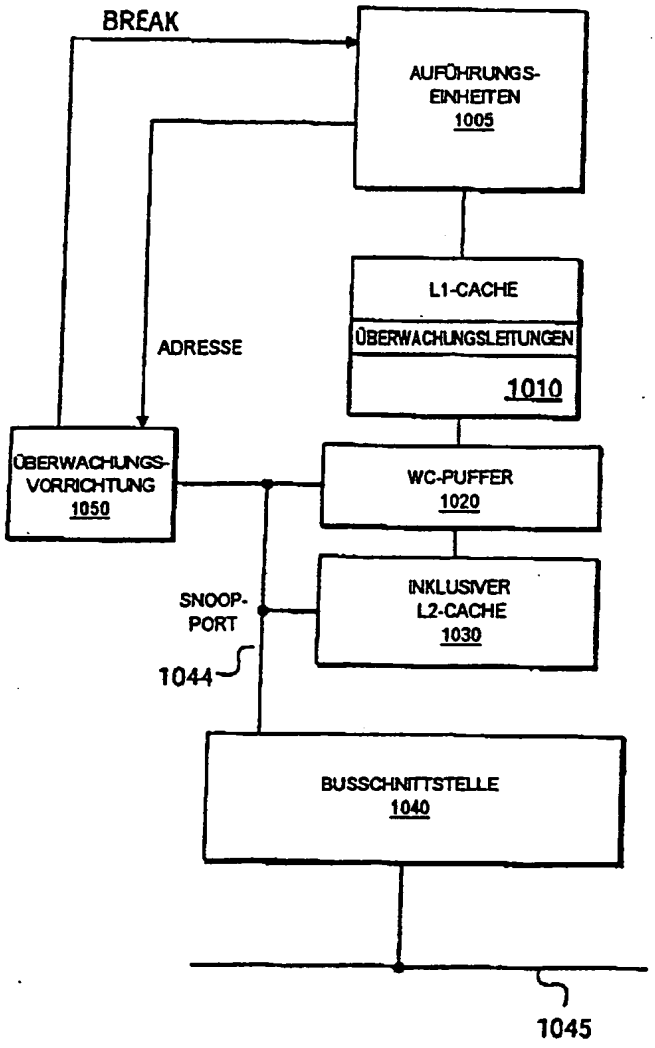


FIG. 10

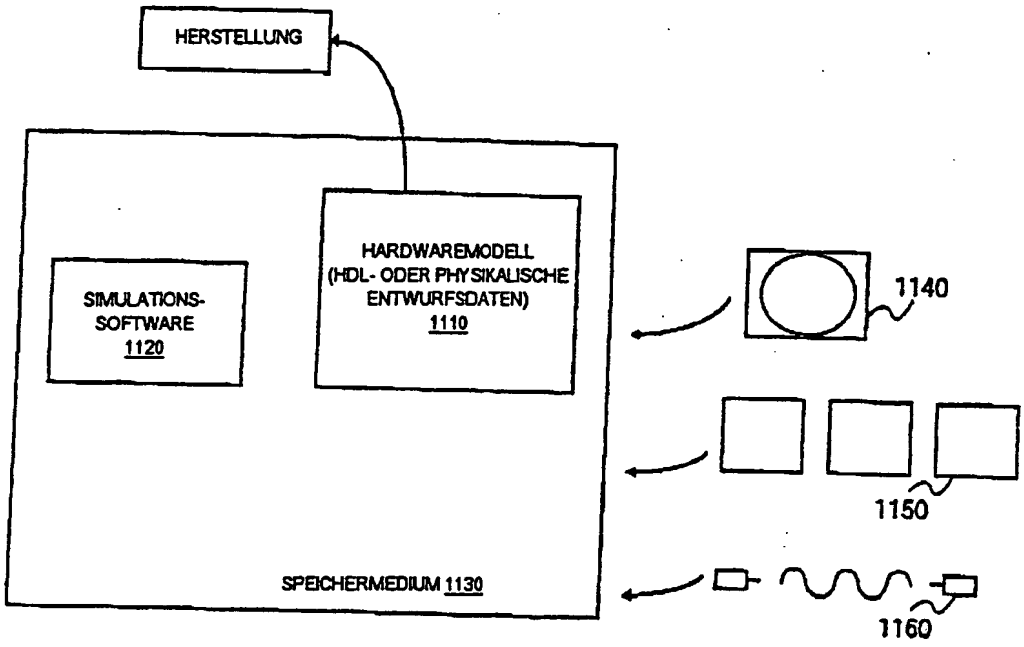

HERSTELLUNG
HARDWAREMODELL
(HDL- ODER PHYSIKALISCHE
ENTWURFSDATEN)
1110
SIMULATIONS-
SOFTWARE
1120
SPEICHERMEDIUM 1130
1140
1150
1160

FIG. 11